

19990115 064

AEA/TYKB/28006/RP/1

3D BODY-FITTED SOFTWARE ON
THE MHPCC SP2 COMPUTER

FINAL REPORT

J W Eastwood, W Arter, N J Brealey, R W Hockney

September 1995

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB

email: jim.eastwood@aeat.co.uk

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data clause at DFARS 252227-7013.
AEA Technology, Culham Laboratory, Abingdon, Oxfordshire, OX14 3DB, England.

Document Control Number: AEA/TYKB/28006/RP/1			
Date of Issue: September 1995			Issue number: 1
<i>Authorization</i>	<i>Name</i>	<i>Signature</i>	<i>Position</i>
Prepared by	J W Eastwood	<i>J. W. Eastwood</i>	Project Manager
Checked by	W Arter	<i>W. Arter</i>	Project Scientist
Approved by	J W Eastwood	<i>J W Eastwood</i>	Project Manager

©— United Kingdom Atomic Energy Authority, 1995 — ©

DDO QUALITY ASSURED 3

AQF99-04-0636

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1995	3. REPORT TYPE AND DATES COVERED Final Report	
4. TITLE AND SUBTITLE 3-D Body-fitted Software on the MHPCC SP2 Computer			5. FUNDING NUMBERS F6170895C0010	
6. AUTHOR(S) Dr James William Eastwood				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Culham Laboratory, UKAEA Abingdon OX14 3DB United Kingdom			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0200			10. SPONSORING/MONITORING AGENCY REPORT NUMBER SPC 95-4019	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (Maximum 200 words) This report results from a contract tasking Culham Laboratory, UKAEA as follows: write software and datasets for a SUN workstation at Phillips Laboratory and on Maui High Performance Computer Center SP2 computer.				
14. SUBJECT TERMS EOARD			15. NUMBER OF PAGES	
			16. PRICE CODE N/A	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

<i>TYKB/28006 Final Report</i>	1
--------------------------------	---

Contents

1 Introduction	3
2 Programme of Work	5
2.1 Subtask 1.1	5
2.2 Subtask 1.2	6
2.3 Subtask 1.3	6
2.4 Subtask 1.4	6
 Annexes	 10
A AEA/TYKB/28006/TN/1	11
B AEA/TYKB/28006/TN/2	23
C AEA/TYKB/28006/TN/3	145
D AEA/TYKB/28006/TN/4	171
E AEA/TYKB/28006/TN/5	197
F CPC 87(1995)155	215
G MAUI-3DPIC/README-MIMD	243
H README-LPM3	253

3D BODY-FITTED SOFTWARE ON THE MHPCC SP2 COMPUTER

FINAL REPORT

J W Eastwood, W Arter, N J Brealey, R W Hockney

September 1995

Abstract

This document, together with the software and test datasets supplied and installed on a SUN workstation at Phillips Laboratory and on the Maui High Performance Computer Center SP2 computer, are the completion of the contract PIIN:F61708-95-C0010.

1 Introduction

This document is the Final Report for the work undertaken under Contract F61708-95-C0010 between EOARD and AEA Technology, Culham. This work is a successor to an earlier project *3-D PIC Software for MIMD Computers*, PIIN:F61708-93-C0011, and the material presented here assumes details of the algorithms and software design given in earlier documents issued during that predecessor contract [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], and elsewhere [15, 16, 17, 18].

Attached to this report are eight Annexes which contain results of the parallel benchmarking, further algorithm analysis, test examples and revised specifications. These Annexes are:

Annex A:

R W Hockney[19],
LPM3 Benchmark Results on the MHPCC IBM SP2 Computer,
Technical Note AEA/TYKB/28006/TN/1,
Culham Laboratory, September 1995.

This Note summarises the implementation and testing of the new message passing routines and compares the parallel performance of different options on the SP2 with each other and with the parallel performance measurements made earlier on the Intel Paragon at Sandia.

Annex B:

J W Eastwood, W Arter, N J Brealey and R W Hockney[20],
Demonstration Calculations of HPM Sources using 3-D Electromagnetic

PIC Software,
Technical Note AEA/TYKB/28006/TN/2,
Culham Laboratory, September 1995.

This Note describes the changes made to produce the Version 2.10.00 of the software issued with this report, and presents test results obtained with this new version.

Annex C:

J W Eastwood, W Arter, and N J Brealey[21],
3DPIC Release 2.10.00: User Guide,
Technical Note AEA/TYKB/28006/TN/3,
Culham Laboratory, September 1995.

This Note contains a description of the user input files (.uif)s, and an introduction to running the preprocessor PEGGIE, the simulation code PIC3D, postprocessors TSDMERGE, MPICTIM and DISPAN, together with the GHOST graphical system.

Annex D:

W Arter[22],
Release 2.10.00 of the PEGGIE Preprocessor,
Technical Note AEA/TYKB/28006/TN/4,
Culham Laboratory, September 1995.

This Note describes the PEGGIE preprocessor and its input specification. The concept underlying the PEGGIE (Parallel Electromagnetic General Geometry Interface) approach to grid generation, boundary conditions, material property specification and diagnostics, is that the device to be modelled and the diagnostic lines, surfaces and volumes are made by joining 'parts' together. An important advantage is that a 'part' might be output of another geometry package (eg [23]), although normally it will consist of a subroutine and the associated variables. The Note extends the specification give in Reference [14], and supersedes that document.

Annex E:

N J Brealey[24],
MPICTIM Release 2.10.00: User Guide,
Technical Note AEA/TYKB/28006/TN/5,
Culham Laboratory, September 1995.

The timeseries output files (.tsd files) produced by PIC3D diagnostics can be examined on a UNIX Workstation using the OSF/Motif GUI tool MPICTIM. This Note describes how to use Version 2.10.00 of MPICTIM and supersedes Technical Note AEA/TYKB/31878/TN/10 [12].

Annex F:

J W Eastwood, W Arter, N J Brealey and R W Hockney[18],
Body Fitted Electromagnetic PIC Software for Use on Parallel Computers,
Comput Phys Commun 87(1995)155.

This Annex reproduces the paper published in Computer Physics Communications describing the PIC3D algorithm and software.

Annex G:

R W Hockney,
MIMD Version PIC3D Operating Instructions - Maui HPCC,
August 1995.

This Annex is a copy of the "readme" file held in the file
~hockney/MAUI-3DPIC/README-MIMD
on the MHPCC SP2 computer.

Annex H:

R W Hockney,
LPM3 Benchmarking,
September 1995.

This Annex is a copy of the "readme" file held in the file
~hockney/SP2/LPM3/test/README-LPM3
on the MHPCC SP2 computer.

2 Programme of Work

The objectives of the work, as described in the Proposal [25] and restated in the contract document for Contract Number F61708-95-C0010, were:

Subtask 1.1 Implement and test new message passing routines for use on the SP2.

Subtask 1.2 Perform LPM3 benchmarking on the SP2.

Subtask 1.3 Develop PIC3D kernel for the SP2.

Subtask 1.4 Prepare final report.

These Subtasks are discussed in the following Subsections and in the Annexes to which those Subsections refer.

2.1 Subtask 1.1

The message passing software was developed in parallel with the physics kernel of the main simulation program PIC3D (see below). The testbed for the developing message passing routines was the benchmarking version of the code (LPM3). LPM3 is a fully functional 3-D code with the same data structures and main calculation routines as PIC3D, but with limited physics and restricted geometric capabilities.

Two different versions of LPM3 were developed; one using a 'per-patch' message data exchange, and one using a 'per-node' exchange. Both cases were implemented using four different communications interfaces on the SP2 - PVM3ip, PVM3sw, PVMe and MPL (cf Annex A).

The source for the LPM3 benchmark code is installed on the SP2 in directory

`~hockney/SP2/LPM3/pvm3`

Test benchmark runs should be carried out in directory

`~hockney/SP2/LPM3/test`

which contains the input data files for 8, 64, 512 and 4096 block cases. Amongst the files in that directory is the file `README-LPM3` which gives instructions on how to use LPM3. A copy of that "readme" file is attached to this Report in Annex H.

2.2 Subtask 1.2

A series of benchmarking computations have been performed to identify the best mode of operation for PIC3D on the SP2. Our tests show that the native IBM message passing library gives the best performance. However, the speed gain of the IBM customised version of PVM, PVM_e, is almost as good, and has the advantage of maintaining code portability. It is for this reason that we have chosen to use the PVM library for the PIC3D implementation described below.

The public domain version of PVM 3.0, PVM3_{ip}, gives considerably poorer performance, showing slow-down rather than speed-up in some cases as the number of processors is increased. The fourth option, PVM3_{sw}, gave better performance than PVM3_{ip}, but inferior to PVM_e and MPL.

A summary of the SP2 benchmarking results and a comparison with figures on the Intel Paragon at Sandia are given in Annex A.

2.3 Subtask 1.3

LPM3 omits much of the physical complexity of the full 3DPIC code and concentrates on the code necessary for parallel execution and performance measurement. In this subtask the main simulation code PIC3D was advanced to the next stage of its development cycle, and was blended with the parallel code from LPM3 to give the version of PIC3D which operates on either a workstation or parallel computer. A number of refinements have been made to the simulation code algorithm, and several demonstration calculations have been undertaken on both the SUN Workstations at Culham and on the MHPCC SP2 computer.

Details of the code refinements and a number of demonstration calculations are presented in Annexes B.

2.4 Subtask 1.4

This document and its Annexes, together with the software and test data installed on the SUN workstation `ppws20` at Phillips Laboratory under username `eastwood` and on the MHPCC SP2 under the username `hockney` comprise Subtask 1.4 and are the Final Deliverables of this contract.

Installed on the `ppws20` workstation in directory `~eastwood/bin-sun` are the following executables:

Executable	Purpose
peggie	input preprocessor
pic3d	main simulation program
mpictim	time series data postprocessor
dispan	dispersion analysis postprocessor
tsdmerge	merge .tsd files from parallel machine

The source for PIC3D is held in the directory 3DPIC/PIC3D and support libraries for the main simulation program are in the other directories under 3DPIC. The directory 3DPIC/TEST contains the .uif files for the example cases described in the Technical Note contained in Annex B. In addition, the GHOST graphics library was previously installed for displaying, postprocessing and printing graphical output. A number of test datasets have been provided – these test cases are described in Annex B. The preprocessor **peggie** and postprocessors **mpictim** and **dispan** were not developed under this research programme, but are supplied in executable form for use with the main simulation code **pic3d**.

Installed on the SP2 are the source and parallel executables for the main simulation program PIC3D and the benchmark program LPM3. Also provided on the SP2 is the additional postprocessor **tsdmerge** for merging the timeseries output (*.tsd) from the parallel run into a single *.tsd file for transfer to the workstation for postprocessing. The executable **tsdmerge** is held in directory ~hockney/bin. The parallel version of PIC3D is in the file ~hockney/pvm3/bin/RS6K/pic3d. The source for the PIC3D code is in the directory MAUI-3DPIC. For further details see Annexes G and H.

References

- [1] J W Eastwood, W Arter, N J Brealey and R W Hockney, *The 3-D General Geometry PIC Software for Distributed Memory MIMD Computers: Task 1 Final Report*, Culham Laboratory Technical Note AEA/TLNA/31878/RP/1, September 1994.
- [2] J W Eastwood, W Arter and R W Hockney, *The 3-D General Geometry PIC Software for Distributed Memory MIMD Computers: EM Software Specification*, Technical Note AEA/TLNA/31878/TN/1, Culham Laboratory, January 1994.
- [3] J W Eastwood, *The 3-D General Geometry PIC Software for Distributed Memory MIMD Computers: Data Organisation*, Technical Note AEA/TLNA/31878/TN/2, Culham Laboratory, January 1994.
- [4] R W Hockney, *The 3-D General Geometry PIC Software for Distributed Memory MIMD Computers: Patch Exchange Tables and Subroutines*, Technical Note AEA/TLNA/31878/TN/3, Culham Laboratory, January 1994.
- [5] W Arter, *The 3-D General Geometry PIC Software for Distributed Memory MIMD Computers: Description of Inputs Used to Generate Meshes*,

- Technical Note AEA/TLNA/31878/TN/4, Culham Laboratory, January 1994.
- [6] J W Eastwood, W Arter, N J Brealey and R W Hockney, *Body Fitted PIC Software for Microwave Device Modelling*, EUROEM, Bordeaux, France May 1994.
 - [7] W Arter, J W Eastwood, N J Brealey and R W Hockney, *Electromagnetic Modelling in Arbitrary Geometry by PIC Methods on MIMD Computers*, pp 297-300 in 6th Joint EPS-APS Int Conf on Phys Computing PC'94 (R Gruber and M Tomassini, eds), EPS, Geneva(1994).
 - [8] R W Hockney, *LPM3 Benchmark Results on the Intel Paragon and iPSC/860*, Technical Note AEA/TYKB/31878/TN/6, Culham Laboratory, July 1994.
 - [9] W Arter, *The System of Dimensionless Units*, Technical Note AEA/TYKB/31878/TN/7, Culham Laboratory, September 1994.
 - [10] W Arter, *Dispersion and Stability Analysis for Maxwell's Equations*, Technical Note AEA/TYKB/31878/TN/8, Culham Laboratory, September 1994.
 - [11] N J Brealey, J W Eastwood and W Arter, *3DPIC Diagnostics*, Technical Note AEA/TYKB/31878/TN/9, Culham Laboratory, September 1994.
 - [12] N J Brealey, *MPICTIM User's Guide*, Technical Note AEA/TYKB/31878/TN/10, Culham Laboratory, September 1994.
 - [13] W Arter, *Boundary Conditions for Maxwell's Equations in General Geometry*, Technical Note AEA/TYKB/31878/TN/11, Culham Laboratory, September 1994.
 - [14] W Arter, *Extended Description of Inputs used to Generate Meshes*, Technical Note AEA/TYKB/31878/TN/13, Culham Laboratory, September 1994.
 - [15] J W Eastwood, *The Virtual Particle Electromagnetic Particle-Mesh Method*, Computer Phys Commun 64(1991)252-266.
 - [16] J W Eastwood, R W Hockney and W Arter, *General geometry PIC for MIMD computers*, Report RFFX(92)52, Culham Laboratory, August 1992.
 - [17] J W Eastwood, R W Hockney and W Arter, *General geometry PIC for MIMD computers: Final report*, Report RFFX(93)56, Culham Laboratory, June 1993.
 - [18] J W Eastwood, W Arter, N J Brealey and R W Hockney, Comput Phys Commun, 87(1995)155.
 - [19] R W Hockney, *LPM3 Benchmark Results on the MHPCC IBM SP2 Computer*, Technical Note AEA/TYKB/28006/TN/1, Culham Laboratory, September 1995.

- [20] W Arter, *Boundary Conditions for Maxwell's Equations in General Geometry*, Technical Note AEA/TYKB/31878/TN/11, Culham Laboratory, September 1994.
- [21] W Arter, *Extended Description of Inputs used to Generate Meshes*, Technical Note AEA/TYKB/31878/TN/13, Culham Laboratory, September 1994.
- [22] W Arter, *The 3-D General Geometry PIC Software: The Current State of the PEGGIE Preprocessor*, Technical Note AEA/TYKB/31777/TN1, Culham Laboratory, May 1995.
- [23] J W Eastwood, W Arter, N J Brealey and R W Hockney, *Comput Phys Commun*, 87(1995)155.
- [24] W A J Prior, *GHOST USER MANUAL Version 8*, UKAEA Culham Laboratory, ISBN 0-85311-184-7, 1991.
- [25] R W Hockney, *LPM3 Benchmark Results on the MHPCC IBM SP2 Computer*, Technical Note AEA/TYKB/28006/TN/1, Culham Laboratory, September 1995.
- [26] J W Eastwood, W Arter, N J Brealey and R W Hockney, *Demonstration Calculations of HPM Sources using 3-D Electromagnetic PIC Software*, Technical Note AEA/TYKB/28006/TN/2, Culham Laboratory, September 1995.
- [27] J W Eastwood, W Arter, and N J Brealey, *3DPIC Release 2.10.00: User's Guide*, Technical Note AEA/TYKB/28006/TN/3, Culham Laboratory, September 1995.
- [28] W Arter, *Release 2.10.00 of the PEGGIE Preprocessor*, Technical Note AEA/TYKB/28006/TN/4, Culham Laboratory, September 1995.
- [29] N J Brealey, *MPICTIM Release 2.10.00: User Guide*, Technical Note AEA/TYKB/28006/TN/5, Culham Laboratory, September 1995.

ANNEXES

A AEA/TYKB/28006/TN/1

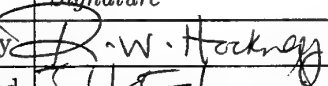


AEA/TYKB/28006/TN/1

THE 3-D GENERAL GEOMETRY PIC
SOFTWARE: LPM3 BENCHMARKING
ON THE MHPCC SP2

Roger W. Hockney

September 1995

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB

Document Control Number: AEA/TYKB/28006/TN/1			
Date of Issue: September 1995			Issue number: 1
<i>Authorization</i>	<i>Name</i>	<i>Signature</i>	<i>Position</i>
Prepared by	R W Hockney		Project Scientist
Checked by	J W Eastwood		Project Manager
Approved by	J W Eastwood		Project Manager

Contents

1 Background	3
2 LPM3 Benchmark	3
3 IBM SP2 Communication Alternatives	4
4 Comparison of Paragon and SP2	5
5 Conclusions	6

THE 3-D GENERAL GEOMETRY PIC SOFTWARE: LPM3 BENCHMARKING ON THE MHPCC SP2

Roger W. Hockney

September 1995

Abstract

The impact of parallelisation issues on the 3DPIC software has been studied using the LPM3 benchmark. Successful studies have been completed on both the Intel Paragon (with up to 1840 processors) and the IBM SP2 (with up to 128 processors). The best option for maintaining portability and getting speedup on the SP2 is the per processor code using PVMe message passing routines.

1 Background

In order to demonstrate the parallel performance of the new 3D MIMD PIC code (3DPIC) the benchmark version of the code (called LPM3) has been run on the Intel Paragon at the Sandia National Laboratory, Albuquerque, and on the IBM SP2 at the Maui High Performance Computer Center (MHPCC). This extends the previously reported results for the Paragon and the Intel iPSC/860.

The IBM SP2 offers several different communication interfaces, and four of these are compared: Public domain PVM3 (using ethernet or switch), IBM custom PVMe, and the IBM Message Passing Language (MPL). For each interface the performance of both the per-patch and per-processor versions of the code have been measured.

2 LPM3 Benchmark

The LPM3 benchmark code simulates a triply-periodic three-dimensional electron plasma. The plasma space is divided into blocks, each of which contains 512 particles representing the electrons, and 64 elements on which the fields are calculated. From the point of view of load balancing on the parallel computer, the block is the smallest unit that can be allocated amongst the processors. The problem size is measured by number of blocks N_b , and four problem sizes have been used with respectively $N_b = 8, 64, 512, 4096$. These correspond respectively to numbers of particles $N_p = 4K, 32K, 256K, 2M_2$ where $K = 1024$ and $M_2 = K^2$. The timestep is such that about ten percent of the particles leave each block and enter neighbouring blocks during a timestep. A run of 100

timesteps is chosen as the benchmark test because this can be done in a few minutes for problem sizes and number of processors of interest (tstep/s in the range 0.1 to 10), and the conservation of total number of particles is used as a validity check.

There are two different versions of the benchmark code, which are selected by the value of the last input variable MXPASW. The per-patch version sends a separate message for every patch in the system, and there are 9 patches for every block. In the per-process version, on the other hand, the patch messages are assembled and sorted in a buffer so that only one message is sent to every other process to which a given process is attached. The per-process code may send 10 or 100 times fewer messages than the per-patch version. The per-process version should be significantly faster than the per-patch version on computers with a high message startup time or latency. For computers with low latency there will be little difference between the versions.

3 IBM SP2 Communication Alternatives

The IBM SP2 offers the following communication interfaces:

PVM3ip: Public domain PVM version 3.0 from the Oak Ridge National laboratory with full Internet communication protocol (ip) between processors, using ethernet connections. It may be used to connect processors of different types, as in a cluster of heterogeneous workstations. Word format conversion between different types of processor is accommodated. For these reasons PVM3 has a high startup overhead, but it is implemented on all parallel computers of importance. PVM3 code should therefore be highly portable. (Data was taken at MHPCC on a dedicated partition of 64 MByte thin nodes on 4 January 1995).

PVM3sw: Same as above, but using IBM high-speed switch instead of ethernet connections. (Data was taken at MHPCC on a dedicated partition of 64 MByte thin nodes on 4 January 1995).

PVMe: IBM customised version of PVM. This assumes the processors are all the same and within a single SP2. Most of the internet protocol can be omitted and no word format conversion is needed. The IBM high-speed switch is used. The source code is the same as PVM3 and therefore portable. (Data was taken at MHPCC on a dedicated partition of 64 MByte thin nodes on 7 March 1995, using PVMe 1.3.1, xlf 3.2, poe 1.2.1 under AIX 3).

MPL: This is the "native" IBM SP2 Fortran library of communication subroutines. This provides minimum startup overhead in a high-level language. However MPL is unique to IBM, and MPL code is therefore not portable to other computers. The high-speed switch is used. (Data was taken at MHPCC on a dedicated partition of 64 MByte thin nodes on 7 March 1995, using xlf 3.2, poe 1.2.1 under AIX 3).

In order to compare these alternatives we show in Fig. 1 the performance of both versions of the LPM3 benchmark on the 8-block case. The symbol plotted identifies the version used, and the type of dotted line identifies the communication interface. The graph plots Temporal performance in timestep per second (tstep/s) against the number of processors used on a log/log scale, in order to show the scaling of performance with the number of processors. The ideal linear scaling, in which the performance is directly proportional to the number of processors, is shown by the dotted line at 45 degrees.

Public domain PVM3 performs badly with this code, with the per-patch version showing no speedup over single-processor performance with either PVM3ip or PVM3sw (lowest pair of curves). The per-processor version shows a distinct improvement (next higher pair of curves), but the speedup scarcely improves with more than two processors, and is at most two for eight processors (PVM3sw). In both cases use of the switch (sw) is slightly better than ethernet (ip) but the difference is not significant. The advantage of combining the patch messages into a single longer message for each connected processor is clearly seen.

The next higher pair of curves are for IBM PVMe with the per-processor version having the better performance for all processor numbers. The step-like appearance of these curves is due to load imbalance between the processors. For 1, 2, 4 and 8 processors, each processor has the same number of blocks, namely 8, 4, 2 and 1 block, and therefore the same work to do. The load is exactly balanced across the processors, and the performance is optimal. In other cases the load is unbalanced and the performance is determined by the processors with the largest number of blocks. Other processors will have to wait for these critical processors to finish. For 3 processors the largest number of blocks per processor is 3, which explains why the 3 processor performance lies between that for 2 processors (4 blocks per processor) and that for 4-processors (2 blocks per processor). For 5, 6, and 7 processors, there is always at least one processor with two blocks, which explains why the performance for these three cases is almost the same as that for 4 processors (2 blocks per processor). The measured performance for IBM MPL follows the pattern as for PVMe but is in most cases about 10 to 20 percent better. For both PVMe and MPL, there is little difference between the performance of the per-patch and per-processor versions, showing that the message startup time is sufficiently small that reducing the number of messages sent in the per-processor version produces little saving in time.

4 Comparison of Paragon and SP2

In order to compare the performance of LPM3 on the Intel Paragon with the IBM SP2, we show in Fig. 2 the performance of the per-processor code, which was the best on both computers, for the 4 cases of 8, 64, 512 and 4096 blocks. The Intel Paragon data is for the NX2 communication library running under the Sandia SUNMOS 1.4.8 operating system which may be considered to be the Intel equivalent of the native IBM MPL interface. These results have been

previously reported. Although we have measured both versions on the IBM SP2 under both MPL and PVMe, we plot only the per-processor results which were consistently better than the per-patch results. The plotting symbol used distinguishes the problem size (number of blocks), and the line type the computer and communication combination.

Looking at the performance for one processor we find the IBM SP2 to be about 2.5 times faster than the Intel Paragon, and this difference in single processor performance determines the other results. The dotted lines at 45 degrees show the perfect linear scaling from the one-processor performance for the 8-block case. This assumes that performance is directly proportional to the number of processors and inversely proportional to the number of blocks. The Paragon scales extremely well right up to 1840 processors which is the maximum available, and this enables it to overcome the poorer single-processor performance in the 512 and 4096 block cases, although there is not enough parallelism in the 64 and 8-block cases for this to be possible, because one cannot use more processors than there are blocks. The SP2 performance shows a significant fall off from the ideal scaling for more than 16 processors on the 64-block case and more than 64 processors for the 512 block case. Although there are 400 processors on the Maui SP2, the maximum that can be used in practice is 128, so it has not been possible to test the SP2 scaling beyond 128 processors.

5 Conclusions

For the best performance on the IBM SP2 it is advisable to use the native communication interface, but portable PVM code run under IBM customised PVMe is almost as good. For the LPM3 benchmark, public domain PVM3 used over either ethernet (corresponding to a cluster of workstations) or the switch has too much startup overhead to give any worth-while speedup. The significantly better single-processor performance of the SP2 compared to the Paragon, gives the SP2 an initial advantage, but this is lost for the larger problems by the better scaling of the Paragon and the much larger number of processor that are available. PVM code has not been run on the Paragon. The per-processor version of the communication interface has consistently better performance than the original per-patch version, but this difference is only significant for PVM3 communications when it is dramatic.

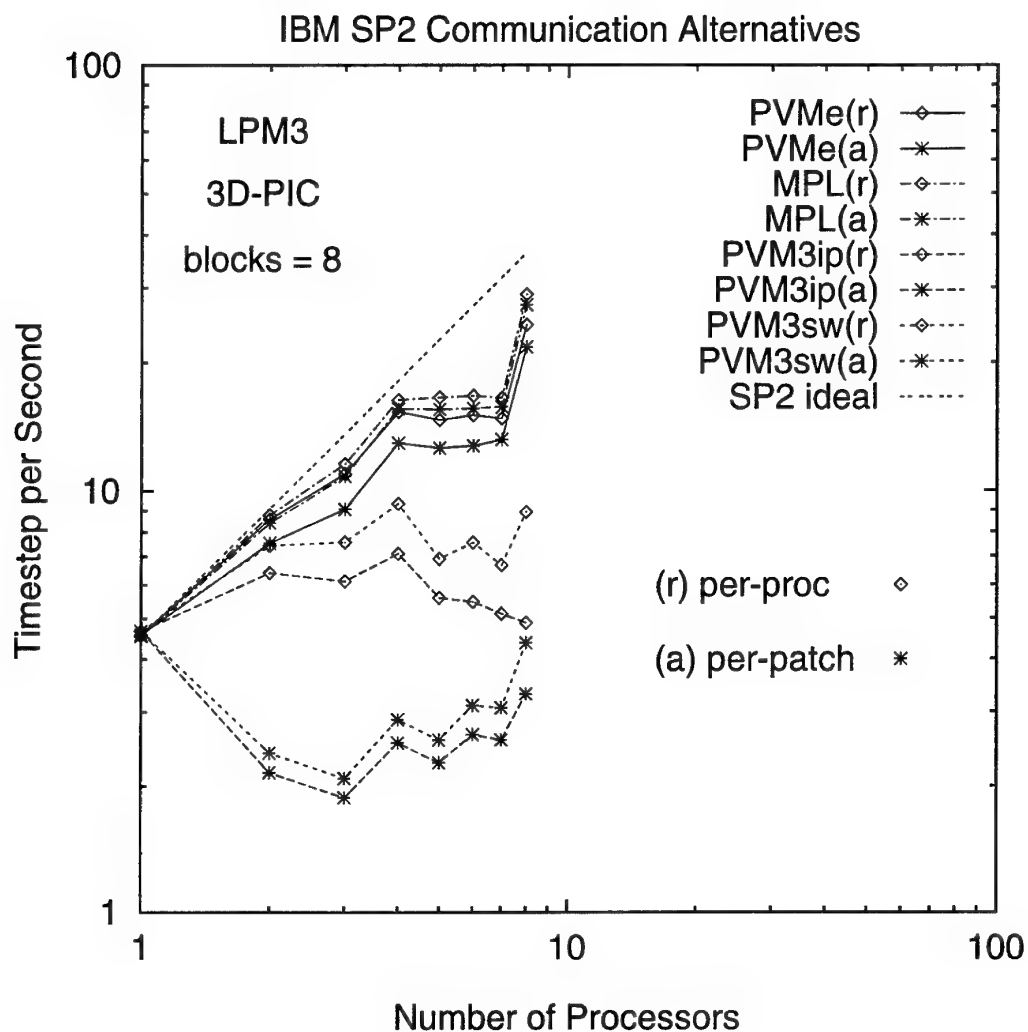


Figure 1: Temporal Performance of the LPM3 benchmark measured in units of timestep per second for the 8-block case on the IBM SP2. Four different communication alternatives are shown: PVM3ip (Public Domain PVM3 using Internet protocol over ethernet connections), PVM3sw (same but over IBM high-speed switch), PVMe (IBM customised PVM over switch) and MPL (IBM's native Message Passing Language over switch). Two code versions are shown: (a) per-patch and (r) per-processor. Line type distinguishes the communication interface, and the plotting symbol the code version.

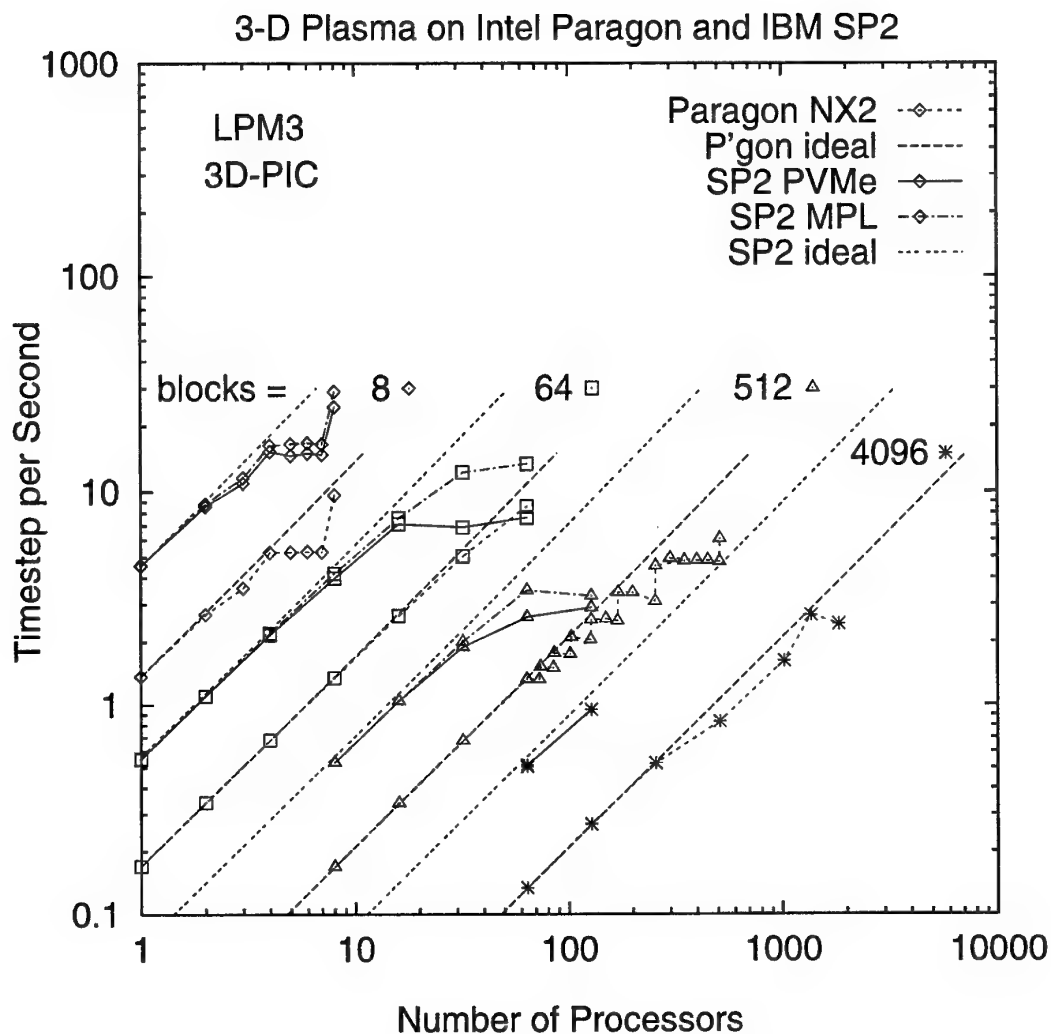


Figure 2: Temporal Performance of the LPM3 benchmark (per-processor version) for four problem sizes on the Intel Paragon and IBM SP2. The plotting symbol distinguishes the problem size (number of blocks) and the line type the computer/communication combination. The step-like shape of the Intel results is due to load imbalance. Except for the 8-block case the IBM results only show cases of exact load balance, and the underlying step-like behaviour is not seen.

B AEA/TYKB/28006/TN/2

AEA/TYKB/28006/TN/2

DEMONSTRATION CALCULATIONS
USING THE 3DPIC SOFTWARE SUITE

J W Eastwood, W Arter, N J Brealey
and R W Hockney

September 1995

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data clause at DFARS 252.227-7013.

AEA Technology, Culham Laboratory, Abingdon, Oxfordshire OX14 3DB, England.

Document Control Number: AEA/TYKB/28006/TN/2			
Date of Issue: September 1995			Issue number: 1
<i>Authorization</i>	<i>Name</i>	<i>Signature</i>	<i>Position</i>
Prepared by	W Arter	W. Arter	Project Scientist
Checked by	J W Eastwood	J. W. Eastwood	Project Manager
Approved by	J W Eastwood	J W Eastwood	Project Manager

Contents

1	Introduction	3
2	The Preprocessor PEGGIE	4
3	The Main Simulation Program PIC3D	5
3.1	Surface Conditions	5
3.2	Particles	9
3.2.1	Particle Equations	9
3.2.2	Particle Dynamics in General Geometry	11
3.2.3	Particle Addressing	14
3.3	Particle Boundary Conditions	15
3.3.1	Particle Emission	15
3.3.2	Particle Beam Injection	17
3.3.3	Patch Buffer Data	19
3.4	Modifying Initial Conditions	22
4	Postprocessors MPICTIM and DISPAN	23
5	Sample Calculations	25
5.1	Connectivity Test une51	26
5.2	Nonorthogonal Test tp06	27
5.3	Oscillations on Axis cy107	28
5.4	Surface Conditions sbc16z	29
5.5	MILO Cavity Dispersion mlp903	30
5.6	Particle Orbit Tests porb21, porb35 and porb40	32
5.7	Orbit Tests porg60h and porg70 in General Geometry	34
5.8	Particle Emission m1076	36
5.9	Particle Beam Injection bean32	37
5.10	Four Cavity MILO m1407	39
5.11	Helix TWT pc11r01	43
	Appendices	51
A	PEGGIE test data examples	51
B	PIC3D test data example	93

DEMONSTRATION CALCULATIONS USING THE 3DPIC SOFTWARE SUITE

J W Eastwood, W Arter, N J Brealey
and R W Hockney

September 1995

Abstract

This Note describes work to enhance the particle modelling capabilities of the 3DPIC suite of codes, and presents a number of sample calculations. The preprocessor PEGGIE, the simulation code PIC3D, output capabilities and the postprocessors MPECTIM and DISPAN have been developed to handle particles in realistic device geometries, and thus to perform and analyse some demonstration electromagnetic and Particle-in-Cell (PIC) calculations.

1 Introduction

Over the past four years, AEA Technology at Culham Laboratory has been developing a new software suite for modelling microwave devices [1]. The software comprises preprocessor, simulation and postprocessor modules. The preprocessor PEGGIE, designed to make the setting up of the simulations simpler and less error prone, provides the capability to prescribe a range of realistic device geometry configurations using a 'kit-of-parts' approach to describing the geometry. The main simulation software PIC3D is built around a general geometry field solver and particle dynamics. The postprocessors, designed to display and measure snapshot and time series output of a number of standard line, surface and volume quantities are at present limited to a subset of electromagnetic quantities and/or specific geometries. These software modules are not yet fully mature, but are sufficiently well advanced to show that they will provide effective tools for computer simulation studies of not only parts of microwave devices, but also (given suitable Massively Parallel Processor Computers) of entire microwave systems with hitherto inaccessible scale lengths.

Reported here are the results of a four month research and development programme to progress the new software tools to the next stage of their development cycle and demonstrate their applicability to HPM source evaluation. A vital part of this process has been the application of the software to real tube design in order to refine the physical models, algorithms and diagnostic outputs of the software. The resulting software provides powerful new capabilities for electromagnetic Particle-in-Cell [2, 3] modelling. It enables three-dimensional (3-D) modelling of microwave tubes and microwave transmission where the

interaction of electromagnetic waves with charged particle flow is important [4, 5, 6], and can equally well be applied to other time dependent problems normally tackled by finite difference, time domain (FDTD) codes.

This report describes work designed primarily to enhance the particle modelling and output capabilities of 3DPIC. The next section (Section 2) treats the modifications made to the PEGGIE preprocessor, then Section 3 explains the changes made to the main PIC3D simulation code, Section 4 deals with the postprocessing, and Section 5 contains sample calculations. Note that the latter Section also demonstrates a model (for a helix Travelling Wave Tube or TWT) produced without using PEGGIE.

2 The Preprocessor PEGGIE

The long term goal of the work is to develop an easy to use 3-D general geometry PIC code for simulating the evolution of electromagnetic fields interacting with charged particle flows in the complex geometries encountered in microwave sources. An essential part is an easy to use preprocessor, one which takes a description of a device and its physics that is comprehensible to the trained engineer, and generates the data needed by the main PIC3D code.

As described in Ref [7], PEGGIE is an interface to a time domain solver for Maxwell's equations. Clearly it has to be extended to initialise PIC calculations. Inspection of the microwave source and transmission modelling problem indicates the need for facilities to:

1. impose a condition on a surface of having a certain impedance together with a spatially varying electric field, to be referred to as the "surface condition",
2. produce line plots of field components as a function of position,
3. plot potential differences as a function of position,
4. simulate the emission of particles from surfaces,
5. produce scatter plots of particle positions.

The incorporation of these changes into PEGGIE has meant the updating of Ref [7] to produce Ref [8]. The need to model particles leads to the introduction of a tag SP to define a species, and the PA tag, which defines patches, is also modified, so that patch particle physics can be defined, as described in Ref [8].

To deal with line plots in a user-friendly fashion, it is possible to define a plotting domain in terms of a curve, and the curve (CU tag) may be defined by inheritance and extension. In other words we can define a curve by saying where it runs in a given block, then demanding that it be extended through some user defined set of blocks called the "universe". It was decided that curves should be calculated in terms of the co-ordinates wherein each block is a unit cube. This has the advantage of being independent of the local meshing, and the conversion to the element based data used to generate the 3DPIC diagnostics is straightforward. While line extension in principle is straightforward, because

PEGGIE allows complex block topologies, problems may arise at block edges, due to the ambiguity about which adjacent block is to be sampled. Unless these are resolved, loop-like curves may not close properly.

The obvious way to resolve this difficulty is to try to duplicate an element based model. In essence, the element based approach removes ambiguities by mapping points onto a lattice which connects unambiguously to adjacent blocks. Our approach is lattice-free, but the idea of having a near-by point which belongs to a definite glue patch and therefore enables unambiguous connection, is taken up. Such a near-by point is called a “fuzz-point” : we choose to generate fuzz-points by mapping local co-ordinates, $\xi_i \in [0, 1]$ to $\eta_i = 2(\xi_i - 1/2)$ and calculating η'_i by reducing the absolute value of η_i by 0.1% for each i on the block surface to give $\xi'_i = (1 + \eta'_i)/2$. Thus provided no very tiny blocks are defined, if the original ξ_i sits on the edge of a glue patch, the fuzz-point ξ'_i will not. Once the connectivity is established, the point corresponding to ξ_i in the adjacent block is calculated as exactly as real floating-point arithmetic allows. The only ambiguity emerges when ξ_i is at the centre of a block face, ie (0,0) in the η_i system. This is resolved by using branch directions, which essentially record the sign of $(\eta_i - \eta'_i)$ for the previous block boundary crossing, and use this to determine the new η_i .

Facility (2) for plots of potential against position means essentially plots of $\int \mathbf{E} \cdot d\mathbf{l}$ against a curve normal to $d\mathbf{l}$. Here, the user is invited to define secondary sub-domains (explained in Ref [9]) for himself using the SD tag. It is helpful to note that these sub-domains must also be two-dimensional and in general should be full joins with the primary domains, as this is assumed in the diagnostic collection routines of PIC3D.

Facility (5) to trigger scatter plot production, is enabled by amending the FL tag, to allow a sampling parameter to be set. This prevents excessive numbers of particles appearing.

The checking of the new facilities is performed using the input datasets or .uif listed in Appendix A. Testing is now a two stage process, since PEGGIE is no longer simply generating geometrical information which can be inspected graphically, but complete datasets (or .dat files) for input to PIC3D. Thus the .dat files have first to be generated, then PIC3D run to validate the original .uif files. Hence the output of the test calculations as described in Section 5 provides the validation for the changes to PEGGIE.

3 The Main Simulation Program PIC3D

The first three subsections of this Chapter describe changes made to the main simulation program. The final subsection contains analytical results that yield an improved initial field condition.

3.1 Surface Conditions

To treat surface conditions requires some further notation, in addition to that used to treat resistive walls in Ref [10]. Equation (3.25) of Ref [10] becomes

Table 1: Entries in LBLAS

Position	Quantity	Related Field(s)
1	INCP(1)	$\alpha_k, \beta_k, \gamma_k$
2	INCP(2)	$\alpha_k, \beta_k, \gamma_k$
3	ISPACE	$\alpha_k, \beta_k, \gamma_k$
4	IORTHO	
5	INCP(1)	d_{ext}
6	INCP(2)	d_{ext}
7	ISPACE	d_{ext}
8	null	
9	INCP(1)	γ_{1i}
10	INCP(2)	γ_{1i}
11	ISPACE	γ_{1i}
12	null	
13	INCP(1)	γ_{2i}
14	INCP(2)	γ_{2i}
15	ISPACE	γ_{2i}

$$\mathbf{e}^3 \times (Z\mathbf{I}_s - [\mathbf{E} - \mathbf{E}_{ext}]) = \mathbf{0} \quad (1)$$

whence it follows that

$$f_3^3 = -(E^3 - E_{ext}^3) \quad (2)$$

$$= -\frac{(d^3 - d_{ext}^3)}{\epsilon_0 \sqrt{g}} \quad (3)$$

Consequently, the equivalent of (4.3) is

$$I^k = \frac{\sqrt{g^{33}}}{\epsilon_0 Z} ([d^k - d_{ext}^{k(n+1/2)}] - \frac{g^{k3}}{g^{33}} [d^3 - d_{ext}^3]), \quad k = 1, 2 \quad (4)$$

We then introduce $\chi_k = 2/(\tilde{Z} + 2\theta)$, where $\tilde{Z} = \epsilon_0 Z/(\Delta t \sqrt{g^{33}})$, so that the update Eq (4.5) of Ref [10] remains invariant provided now (if $d_{ext}^3 = 0$ is assumed to avoid possible numerical instability)

$$\gamma^k = \gamma_{ki} d_i + \chi_k d_{ext}^{(n+1/2)} \quad (5)$$

Note that χ_k depends on k because I^1 and I^2 are defined at different locations and \tilde{Z} may depend on position. Hence it is necessary to define how scalar data are to be stored on surface patches, since in particular this permits a compact storage for the case when \tilde{Z} is position independent.

The three-dimensional addressing idea used in Ref [12] is modified to

$$\text{LOC} = 1 + I \times \text{INCP}(1) + J \times \text{INCP}(2) \quad (6)$$

Table 2: Entries in BCATR

Position	Entry
1	2
2	NSIG
3	ISURF
4	d_{ext}^1
$N_d + 4$	d_{ext}^2
$2N_d + 4$	2
$2N_d + 5$	α_1
$N_{ab} + 2N_d + 5$	α_2
$2N_{ab} + 2N_d + 5$	β_1
$3N_{ab} + 2N_d + 5$	β_2
$4N_{ab} + 2N_d + 5$	χ_1
$5N_{ab} + 2N_d + 5$	χ_2
$6N_{ab} + 2N_d + 5$	γ_{11}
$N_\gamma + 6N_{ab} + 2N_d + 5$	γ_{12}
$2N_\gamma + 6N_{ab} + 2N_d + 5$	γ_{21}
$3N_\gamma + 6N_{ab} + 2N_d + 5$	γ_{22}

where INCP is an array of increments, entries of which will be zero if there is no dependence on the corresponding patch co-ordinate. LOC=1 corresponds to the 'O' point of a block and the patch co-ordinates are directed from the 'O' point towards the 'X' point. The preferred way of ordering the data is to have the patch 1 co-ordinate information stored contiguously. However, difficulties were encountered in routine BCOPAT of PIC3D, and a strategy allowing $INCP(1) \geq INCP(2)$ was adopted as an interim measure. Subroutine MASKSA was written to test the dependence of scalar field s on position, SQASHS to perform any resulting compressions and SCADR to set up INCP and the rest of the patch addressing structure. The addressing structure for patch IPATCH starts at entry LOBLAS (NBTYPE + IPATCH) of the array LBLAS. The entries in relative position are given in Table 1.

Since they vary only with \tilde{Z} , the quantities α_k, β_k and χ_k have a common spatial dependence. Note that ISPACE is the total space occupied by a field and IORTHO is the orthogonality key, denoting whether the γ_{ki} are present. The coefficients are stored in array BCATR starting at location MEMBCA (IPATCH) + 1, in the order shown in Table 2. NSIG is a number defining the time dependence of d_{ext} ; N_d, N_{ab} and N_γ indicate total scalar storage in the obvious way, and if ISURF is non-zero d_{ext} is defined using the initial vector potential \mathbf{A} , as will now be explained.

The idea is to initialise \mathbf{b} globally, using existing subroutines and variables, calculate \mathbf{H} and finally $\nabla \times \mathbf{H}$, then use that to initialise \mathbf{d} . Suppose that it

is critical to have a certain amplitude of \mathbf{E} or potential difference. If Z_m is the mode impedance so $\mathbf{E} = Z_m \mathbf{H}$, it follows (in vacuo) that

$$\mathbf{E} = Z_m \mathbf{B} / \mu_0 \quad (7)$$

Making \mathbf{E} , Z and \mathbf{B} dimensionless gives

$$\mathbf{E} = Z \mathbf{B} \quad (8)$$

where here and in the remainder of this subsection, we use dimensionless units.

Thus if \mathbf{E}_0 , initial (dimensionless) \mathbf{E} is known, the initial dimensionless \mathbf{B} needs to be set to \mathbf{E}_0/Z . The second complication arises from the assumption that $\mathbf{E} = \mathbf{E}_0 \sin \omega t$, where ω is the mode frequency, so \mathbf{E} vanishes at $t = 0$. However, considering the discrete relation

$$\mathbf{E}_{n+1} - \mathbf{E}_n = \nabla \times \mathbf{H} \quad (9)$$

implies ($t_n = [n - 1/2]$) that

$$2 \sin(\omega/2) \mathbf{E}_0 = \nabla \times \mathbf{H} \quad (10)$$

Hence

$$\mathbf{E}_0 = \frac{1}{2 \sin(\omega/2)} \nabla \times \mathbf{H} \quad (11)$$

The third complication is caused by the fact that $\nabla \times \mathbf{H}$ is not fully assembled at a boundary. This is corrected for by doubling, since $\nabla \times \mathbf{H}$ may be assumed to be symmetric about the boundary (if it is antisymmetric then $\mathbf{d} = \mathbf{0}$).

There is also the identity for $z = 0$ that

$$\sin \omega t = \frac{1}{2} (\sin[\omega t + kz] + \sin[\omega t - kz]) \quad (12)$$

which is another way of saying that d_{ext} generates a forward and backward wave. Thus the total factor to apply to \mathbf{d}_0 is

$$\frac{2}{Z \sin(\omega/2)} \quad (13)$$

where, in terms of the dimensional frequency, $\omega = \omega_d \Delta t = 2\pi f \Delta t$. This frequency factor is computed in PIC3D.

More complications arise because the amplitude of the initial b^i field (not \mathbf{B} itself) is the quantity directly specified via input quantities CMODE or BUNI. To correct for this, routine PHYVEC was added to compute \mathbf{B} from the initial b^i .

PHYVEC is essentially a diagnostic routine, and PIC3D operates the convention that all diagnostics are element based. Even so, the drawback of using the relation

$$\mathbf{B} = b^i \mathbf{e}_i / \sqrt{g} \quad (14)$$

is the need to compute \sqrt{g} . We work instead with the formula

$$\mathbf{B} \cdot \mathbf{e}_j \times \mathbf{e}_k = b^i \quad (15)$$

where (ijk) is a cyclic permutation of (123) , which is treated as a set of three simultaneous linear equations for \mathbf{B} , given b^i . In the element based approach, all quantities in Eq (15) must be regarded as though located at the element centre, so the \mathbf{e}_i and b^i have to be appropriately averaged. The equation solving approach has the advantage that it extends easily to the case where covariant components (H_i) are given, but is rather expensive unless the symmetries of the \mathbf{e}_i are exploited (which is not presently the case). The final step of PHYVEC is to rotate the magnetic field components into the global coordinate system.

Thereafter the maximum \mathbf{B} is compared with the expected value and the ratio CMODE/BUNI is adjusted accordingly. Once the initial \mathbf{B} has an amplitude given by this new CMODE/BUNI, \mathbf{E}_0 can be scaled correctly by division by CMODE or BUNI in the PEGGIE front-end.

A further complication is that the boundary may be at a node, where $d^i = 0$, of the initial field. To prevent this, the option to make a phase shift of an angle RPHSHF(j) to the initial conditions, was added. Entry $j = 1, 2, 3$ corresponds to (r, θ, z) or (x, y, z) in the global co-ordinate system. Thus the fields eg. computed by subroutine TMREC have factors of the form

$$\sin(x + \text{RPHSHF}(1)) \dots \cos(z + \text{RPHSHF}(3)) \quad (16)$$

3.2 Particles

3.2.1 Particle Equations

Particle momenta and positions are stored in terms of their curvilinear coordinates local to the block which contains them. The momentum coordinates are updated in two stages. First, the momentum components are updated at the old particle position, then are transformed to components measured at the new position; this eliminates the explicit appearance of the Christoffel symbol terms.

The established centred time approximation [2, 3, 13] is used for the momentum update at the old particle position. Covariant momentum components k at time level $n + 1/2$, $\bar{p}_k^{(n+1/2)}$, measured at the current particle positions $\bar{x}^{k(n)}$ are found using

$$\bar{p}_k^{(n+1/2)} - \bar{p}_k^{(n-1/2)} = q\Delta t_p \left(E_k^{(n)} + e_{klm}(\bar{v}^{l(n-1/2)} + \bar{v}^{l(n+1/2)})b^{m(n)}/2 \right) \quad (17)$$

where

$$\bar{v}^l = \bar{p}^l / \gamma m_0 \quad (18)$$

are contravariant velocity components, e_{klm} is the permutation symbol, and the suffix "p" attached to Δt reminds us that the particles may be followed using a timestep larger than that used for the field advance. Equation (17) is solved using the method due to Boris [13]; apply a half electric acceleration, compute the relativistic γ , apply a two stage rotate and then complete the electric acceleration. Fields are evaluated at the particle positions using

$$b^i = \mathbf{b}^{(i)} X_{(i)} \quad (19)$$

$$E_i = E_{(i)} V_{(i)} \quad (20)$$

where the basis functions $X_{(i)}$ and $V_{(i)}$ were defined in Ref. [12].

Positions are updated to new time level $n + 1$ and the momentum components at the new positions are computed using a predictor-corrector scheme to solve

$$\bar{x}^{k(n+1)} - \bar{x}^{k(n)} = \left(\bar{v}^{k(n+1/2)}(\bar{x}^{(n)}) + \bar{v}^{k(n+1/2)}(\bar{x}^{(n+1)}) \right) \Delta t_p / 2 \quad (21)$$

and

$$\bar{p}^{l(n+1/2)}(\bar{x}^{(n+1)}) = \mathbf{e}^l(\bar{x}^{(n+1)}) \cdot \mathbf{e}_k(\bar{x}^{(n)}) \bar{p}^{k(n+1/2)}(\bar{x}^{(n)}) \quad (22)$$

In cartesians, this scheme reduces to the usual Lorentz force leapfrog integration scheme and no iteration is required. In cylindrical coordinates, the predictor-corrector can be replaced by direct matrix inversion in the manner proposed by Boris [13]. The treatment of general nonorthogonal geometry follows in Section 3.2.2.

In addition, we have the current assignment relation, that computes the currents resulting from particle motion and which from Ref [12] is given by

$$\mathbf{I}^i = \int dt \sum_p q_p \dot{\bar{x}}_p^{(i)} W_{(i)}(\bar{x}_p^1, \bar{x}_p^2, \bar{x}_p^3, t) \quad (23)$$

where p denotes particle index, q_p is the charge on particle p , $\dot{\bar{x}}_p^{(i)}$ a measure of its velocity and $W_{(i)}$ is a weight function described in detail in Ref. [12].

Schematically, the particle equations to be solved are for kinematics,

$$\Delta \bar{x}^i = v^i \Delta t_p \quad (24)$$

half electric-field accelerate,

$$\Delta_h p_i = \frac{1}{2} q E_i \Delta t_p \quad (25)$$

and magnetic field rotate,

$$\Delta_{rot} p_i = q \Delta t_p e_{ijk} p^j v^k / \gamma \quad (26)$$

together with the current contribution from a single (super-)particle,

$$\mathbf{I}^i = q v^i \Delta t \bar{W} \quad (27)$$

where \bar{W} is a dimensionless integral of the weight function.

The electromagnetic fields in the main simulation code have been made dimensionless, as explained in Ref [14], therefore it is necessary to convert the new quantities appearing in Eqs (24) to (27). Let us introduce the subcycling parameter N_s so that $\Delta t_p = N_s \Delta t$, and adopt the convention that in the remainder of this subsection, dimensional quantities appear only in brackets (“[...]”). If the dimensionless version of Eq (24) is taken to be

$$\Delta \bar{x}^i = v^i N_s \quad (28)$$

then v^i must be in units of $1/\Delta t$, whence it follows that p^i has dimensions $m_0/\Delta t$ and p_i units of $L^2 m_0/\Delta t$, where L is the lengthscale introduced in Ref. [14]. Equation (25) reduces to

$$\Delta_h p_i = m_E E_i \quad (29)$$

provided

$$m_E = \left[\frac{q E_0 \Delta t \Delta t_p}{2 m_0 L} \right] = \left[\left(\frac{m_e}{m_0} \right) \left(\frac{q}{|e|} \right) \right] N_s \quad (30)$$

and similarly Eq (26) becomes

$$\Delta_{rot} p_i = 2 m_E e_{ijk} p^j b^k / \gamma \quad (31)$$

In these units, Eq (27) becomes

$$\dot{x}^i = \frac{q}{N_s} \Delta \bar{x}^i \quad (32)$$

where q is now the dimensionless charge on a species superparticle

$$q = \left[\frac{q}{\epsilon_0 E_0 L^2} \right] \quad (33)$$

As explained in Ref [8], the user defines each species in terms of the mass $[m_s]$ and charge $[q_s]$ on a single particle, together with the number N_{ps} of such particles to be represented by a superparticle. Thus for a particular species $[m_0] = N_{ps}[m_s]$, $[q] = N_{ps}[q_s]$, and so $q/m_0 = q_s/m_s$. Species 1 consists, by convention, of electrons. The simulation code PIC3D works in terms of the particle definitions for NSPEC different species, that appear in the array SPATR, as shown by Table 3. MPAINC entries are allocated for each species.

Table 3: Entries in SPATR

Position	Quantity	Comments
MONEPS = 1	N_{ps}	number of particles per superparticle
MOCPS = 2	$N_{ps} q_s$	dimensionless charge q per superparticle (may be negative)
MRATPS = 3	$[(m_e/m_s)(q_s/ e)]N_s$	m_E in particle dynamics

3.2.2 Particle Dynamics in General Geometry

It is important to note that Eq (26) is only schematic. The Boris scheme uses a time discretisation of the magnetic field acceleration equation of form

$$\mathbf{p}^+ - \mathbf{p}^- = \frac{q \Delta t}{2 m_0 \gamma} (\mathbf{p}^+ + \mathbf{p}^- \wedge \mathbf{B}) \quad (34)$$

where superfixes '-' and '+' denote values before and after respectively. It is easy to verify that Eq (34) is exactly soluble in terms of

$$\sigma = \mathbf{p}^- + \mathbf{p}^- \wedge \Omega \quad (35)$$

and

$$\Omega = \frac{qB\Delta t}{2m_0\gamma} \quad (36)$$

as

$$\mathbf{p}^+ = \mathbf{p}^- + \frac{2\sigma \wedge \Omega}{1 + \Omega^2} \quad (37)$$

Putting Eqs (35), (36) and (37) into component form shows that g_{ij} , g^{ij} and \sqrt{g} at a point are needed. The obvious point to take is the current particle position, and it is natural to express \mathbf{p} using the basis at $\mathbf{x}^{(n)}$, whilst recalling that \mathbf{p} is leap-frogged with respect to \mathbf{x} , ie. defined at a time $t = (n + \frac{1}{2})\Delta t_p$.

However, it is not sufficient simply to store components of the metric tensor, because of Eq (22) which contains a product of basis vectors at different points. The fundamental geometrical quantities are \mathbf{e}_i , \mathbf{e}^i and \sqrt{g} , whence say g^{ij} can be obtained as $\mathbf{e}^i \cdot \mathbf{e}^j$, etc. The vectors \mathbf{e}_i are already stored in the array ECOV. Unfortunately, \sqrt{g} and particularly \mathbf{e}^i , are not so easy to define at every point from an array as their variation over an element is complicated. The approach adopted is therefore to compute the \mathbf{e}^i and \sqrt{g} at \mathbf{x} from the $\mathbf{e}_i(\mathbf{x})$. It follows that it is necessary to limit \bar{x}^i such that $\bar{x}^i < n_i$, because eg \mathbf{e}_3 is not defined at $\bar{x}^3 = n_3$. (There is no problem near $\bar{x}^i = 0$ because all $\bar{x}^i \in (-1, 1)$ are all mapped to zero by the standard FORTRAN integer truncation routine.) We do not consider the effects of round-off error any further.

The next and most important issue is the economical evaluation of the geometrical coefficients needed in particle pushing, given the \mathbf{e}_i . From this point-of-view the two main properties possessed by the \mathbf{e}_i are invariance under coordinate transformation and having a zero entry. The former makes it unnecessary to recompute $\mathbf{e}^l \cdot \mathbf{e}_{k*}$ (the '*' denotes a basis at a different position) in Eq (22), or indeed to iterate; the latter saves multiplicative operations in the dot products. However, considering first the invariance property, we observe that if there is a dependence only upon one co-ordinate say $\mathbf{e}_1(\bar{x}^1)$, this translates into rational linear dependences on \bar{x}^1 for all the \mathbf{e}^i (in general) so making use of invariance, even in this simple case, is not easy. The only case perhaps worth distinguishing is where all the \mathbf{e}_i are invariant, ie. a regular gridding of a cuboid or parallelepiped. However, this deserves a totally separate routine for then $\mathbf{e}^l \cdot \mathbf{e}_{k*} = \delta_k^l$. So far we have made no use of invariance properties.

The most obvious simplification due to null \mathbf{e}_i components occurs in an orthogonal block, if we assume the local coordinates are chosen parallel to the \mathbf{e}_i . It does not seem unreasonable to assume that $\mathbf{e}_1 \parallel \hat{\mathbf{x}}$, $\mathbf{e}_2 \parallel \hat{\mathbf{y}}$ and $\mathbf{e}_3 \parallel \hat{\mathbf{z}}$ where $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are the unit vectors of the local coordinate system. Clearly, the case of completely general \mathbf{e}_i with all components non-zero must be treated. The question as to whether any other types of \mathbf{e}_i are worth special treatment may be answered by the study of e_{ij} , the matrix whose columns are the components of \mathbf{e}_i with respect to the local coordinate system (so that the

e_{i1} are the components of \mathbf{e}_1 , etc.) The inverse of e_{ij} has the sparsity pattern of the corresponding matrix for the \mathbf{e}^i , hence the only worthwhile intermediate case emerges as the (right) cylinder of general cross-section, for which we shall assume the sparsity pattern

$$e_{ij} = \begin{pmatrix} e_{11} & e_{12} & 0 \\ e_{21} & e_{22} & 0 \\ 0 & 0 & e_{33} \end{pmatrix} \quad (38)$$

For each sparsity pattern we shall have separate routines to compute \mathbf{e}^i , \sqrt{g} , the symmetric inner product that gives g^{ij} or g_{ij} , the general inner product eg. for $\mathbf{e}^i \cdot \mathbf{e}_{k*}$, and the matrix-vector product, eg. $g_{ij}p^j$.

For the general orthogonal case we have

$$\mathbf{e}^1 = (1/e_{11}, 0, 0) \quad (39)$$

$$\mathbf{e}^2 = (0, 1/e_{22}, 0) \quad (40)$$

$$\mathbf{e}^3 = (0, 0, 1/e_{33}) \quad (41)$$

$$\sqrt{g} = e_{11}e_{22}e_{33} \quad (42)$$

$$m_{ij} = \mathbf{x}_i \cdot \mathbf{y}_j = \begin{pmatrix} x_{11}y_{11} & 0 & 0 \\ 0 & x_{22}y_{22} & 0 \\ 0 & 0 & x_{33}y_{33} \end{pmatrix} \quad (43)$$

$$m_{ij}v_j = \begin{pmatrix} m_{11}v_1 \\ m_{22}v_2 \\ m_{33}v_3 \end{pmatrix} \quad (44)$$

For the right cylinder

$$\mathbf{e}^1 = \Delta^{-1}(e_{22}, -e_{12}, 0) \quad (45)$$

$$\mathbf{e}^2 = \Delta^{-1}(-e_{21}, e_{11}, 0) \quad (46)$$

$$\mathbf{e}^3 = (0, 0, 1/e_{33}) \quad (47)$$

$$\sqrt{g} = e_{33}\Delta \quad (48)$$

where

$$\Delta = (e_{11}e_{22} - e_{12}e_{21}) \quad (49)$$

$$m_{ij} = \mathbf{x}_i \cdot \mathbf{y}_i = \begin{pmatrix} x_{11}y_{11} + x_{21}y_{21} & x_{11}y_{12} + x_{21}y_{22} & 0 \\ x_{12}y_{11} + x_{22}y_{21} & x_{12}y_{12} + x_{22}y_{22} & 0 \\ 0 & 0 & x_{33}y_{33} \end{pmatrix} \quad (50)$$

$$m_{ij}v_j = \begin{pmatrix} m_{11}v_1 + m_{12}v_2 \\ m_{21}v_1 + m_{22}v_2 \\ m_{33}v_3 \end{pmatrix} \quad (51)$$

For the general case, there are no simplifications. The \mathbf{e}^i are calculated using :

$$\mathbf{e}^i = \frac{\mathbf{e}_j \wedge \mathbf{e}_k}{\sqrt{g}} \quad (52)$$

where (ijk) are a permutation of (123) , and all operations involve full matrices and vectors. Note that in a block with an underlying cylindrical co-ordinate system $\mathbf{e}^l \cdot \mathbf{e}_{k*}$ is further complicated by the need to transform to a cartesian frame to compute the dot product.

Once the geometrical quantities are defined, accelerating the particles is straightforward, given the replacement of cross products such as $\boldsymbol{\sigma} \wedge \boldsymbol{\Omega}$ by $\sqrt{g}e_{ijk}\sigma^j\Omega^k$, and the use of the identity $g\mathbf{B}^2 = g_{ij}b^ib^j$. For the particle move, a number of schemes are possible, once it is realised that Eq (21) is an approximation to

$$\Delta \bar{x}^i = \int v^i dt = \int \mathbf{v} \cdot \mathbf{e}^i dt \quad (53)$$

$$= \mathbf{v} \cdot \int \mathbf{e}^i dt \quad (54)$$

for a velocity \mathbf{v} which is independent of time. The choice arises because \mathbf{e}^i is typically discontinuous from one cell to another, although it also varies within a cell. An approach that involved performing the \mathbf{e}^i integral as a distance-weighted sum over the initial and final \mathbf{e}^i was tried first. However, in the main, the results were no better than achieved using Eq (22), which is equivalent to replacing the integral (54) by the average of \mathbf{e}^i at the end points. Time unfortunately did not permit the investigation of other quadratures; the discontinuous nature of the \mathbf{e}^i suggests that a cell-by-cell approach might be the best.

Block boundaries are already treated with such an approach. The intersection \bar{x}_j^i of the trajectory with the cell boundary is calculated, and the \mathbf{e}^i at \bar{x}_j^i are used in the computation of the \mathbf{e}^i integral for the corrector step. If, at the end of the corrector phase, there is still a boundary intersection, \mathbf{p} is expressed in terms of the basis at the best estimate for \bar{x}_j^i , and then the momentum triple \mathbf{p}_T (see Section 3.2.3) follows. Thus there is a separate predictor-corrector loop for each block traversed by a particle in one time-step.

Lastly, we mention the approach of using the "straight line in Cartesian geometry", viz. approximations based on :

$$\Delta \mathbf{x} = \int \mathbf{v} dt \quad (55)$$

Suitable formulae for the iterative solution of Eq (55) were derived, but it appeared that their implementation would be more expensive than the "straight line in curved space" approach described earlier. An approximation such as replacing each cell by a parallelepiped might be competitive.

3.2.3 Particle Addressing

As explained in Ref [12], records for a given uniblock IBLOCK are to be found in LPARAS and COORDS, starting at the origins LOPARA(IBLOCK) and LOCOOR(IBLOCK) respectively. The layout of data in the uniblock record in COORDS is given by values in LPARAS. The layout of LPARAS is summarised in Table 4. Numerical values for the index locations are given in Table 5, where note that NOPARA is the symbolic value of the number of entries in an addressing record in LPARAS.

Table 4: Contents of a uniblock record in LPARAS

Location	Entry
MSPEC	number of species
MNPOS	number of position coords stored
MNMOM	number of momentum coords stored
MPNOO	number of particles of species of species 1
MPNOO + 1	number of particles of species of species 2
⋮	
MPNOO + MXSPEC - 1	number of particles of species MXSPEC
MPORO	origin in coordinate record of species 1
MPORO + 1	origin in coordinate record of species 2
⋮	
MPORO + MXSPEC - 1	origin in coordinate record of species MXSPEC

To describe the contents of COORDS, we introduce $INPOS = LPARAS(MNPOS)$, $INMOM = LPARAS(MNMOM)$ and $IPLN = INPOS + INMOM$. Relative to the origin $LOCOOR(IBLOCK)$, entries 1 ... $INPOS$ of COORDS contain the contravariant position coordinates of the first particle, entries $INPOS + 1$... $IPLN$ the covariant momentum coordinates of the first particle, entries $IPLN + 1$... $IPLN + INPOS$ the position coordinates of the second particle, and so on until all the $INSPEC = LPARAS(MPNOO)$ particles of the first species have been treated. The particles of the second species follow, starting at a relative location given by $LPARAS(MPORO+1) = INSPEC * IPLN$ if no gaps are left.

3.3 Particle Boundary Conditions

The patch buffers are used to inject particles into the device, and to transfer them between blocks. First we describe how particles may be injected, then the use of the patch buffers is explained.

3.3.1 Particle Emission

The particle emission algorithm extends the scheme used for two-dimensional problems. The main difference is in the way the charge density Q on the patch is computed. Subroutine QSHONP was written to calculate Q in a hybrid manner, viz. using nearest neighbour accumulation in the directions within the surface and distance (area) weighting in the normal direction. This avoids problems arising at patch boundaries where Q may be underrepresented if volume (area) weighing is used in all three co-ordinates.

In subroutine EMITEL, for each row of elements on the patch, the surface charge in each element is computed from a local application of Gauss' law,

Table 5: Parameters for particle addressing

Name	Value
MSPEC	1
MNPOS	2
MNMOM	3
MPNOO	4
MPNOO + 1	5
⋮	
MPNOO + MXSPEC - 1	11
MPORO	12
MPORO + 1	13
⋮	
MPORO + MXSPEC - 1	19
MXSPEC	8
NOPARA	20

which gives

$$\sigma = d - Q \quad (56)$$

where d is the normal component of d^i (averaged to give a value at the centre of the element surface). Two passes are made, the second interchanging the row and element indices of the first. On each pass, particles representing $\sigma/2$ of charge are emitted using the algorithm devised for two-dimensional emission on each row or column (depending on the direction of the pass).

The emission algorithm thus takes the following form. Let $x = \bar{x}^i$ where i depends on the direction of the pass, then the x -coordinate of the next particle emitted is found by summing the charge σ over the line (row or column) of elements in x until it first amounts to more than one electron. A seed charge q_0 where $0 \leq q_0 \leq 1$ is used to start the process, but only on the first row of the first pass. Thus, if q_N is the charge left over after previous creations (so $q_N = q_0$ at the start), then we form

$$q_T = q_N + \sigma \quad (57)$$

The number of particles emitted per element of the pass i_{em} is, since we work in units where the electron has (negative) unit charge, given by

$$i_{em} = \text{int}(-q_T) \quad (58)$$

where int denotes integer part. When i_{em} is positive, the particles have x coordinate

$$x = x_{el} + (i - q_N)/\sigma, \quad i = 1, \dots, i_{em} \quad (59)$$

where x_{el} is the lowest value of x on the element. If y_{el} is defined correspondingly for the patch direction normal to x , then the particle's y coordinate is set to y_{el} plus a random number r lying between zero and unity. Its initial position normal to the boundary is set to a small random value. Note that i_{em} is not allowed to exceed four to prevent excessive emission during transients,

Particles are thought of as having been born at a time $(1 - r)\Delta t_p$ after the start of the present particle step. Hence they initially acquire momentum through being accelerated for a time $r\Delta t_p$ in the local electric and magnetic fields.

Freshly created particles are introduced into the main calculation by placing them in the particle patch buffers, see the next section. They then appear as though they have just crossed from a neighbouring block.

3.3.2 Particle Beam Injection

We assume that the beam is specified via its current profile $\mathbf{j}(\mathbf{x})$, and the energy of the beam particles (of mass m_0 and charge q) via their voltage V . The statement that a beam is at voltage V , then translates to a particle energy \mathcal{E} where presumably

$$\mathcal{E} = m_0 c^2 + qV \quad (60)$$

and $m_0 c^2$ is the rest mass. However, it follows from special relativity that

$$\mathcal{E} = \gamma m_0 c^2 \quad (61)$$

where

$$\gamma = \left(1 - \frac{v^2}{c^2}\right)^{-1/2} \quad (62)$$

Squaring Eq (61) and using

$$\mathbf{p} = \gamma m_0 \mathbf{v} \quad (63)$$

it follows that

$$\mathcal{E}^2 = m_0^2 c^4 + \mathbf{p}^2 c^2 \quad (64)$$

and hence equating to the square of Eq (60)

$$p'^2 = \frac{p^2}{m_0^2 c^2} = \left(\frac{qV}{m_0 c^2}\right)^2 + 2 \left(\frac{qV}{m_0 c^2}\right) \quad (65)$$

where the factors have been arranged so that each of the three terms is dimensionless.

Eq (65) determines the momentum of each particle in a monoenergetic beam. The beam current density determines how many super-particles need to be injected. The idea is that particles appear on a surface S which is normal to the beam direction and which is represented as the union of a collection of surfaces of finite elements. Suppose one such element has surface vector $\Delta \mathbf{S}$, then the current passing through the surface is $\mathbf{j} \cdot \Delta \mathbf{S}$.

The number of particles to be introduced per time-step in an element is thus

$$N_{in} = \frac{\mathbf{j} \cdot \Delta \mathbf{S} \Delta t}{N_{ps} q_s |e|} \quad (66)$$

where the denominator is the charge on a superparticle. (The value of the right-hand side is rounded up or down by the addition of r , a random number in the unit interval.)

In practice \mathbf{j} is normalised using the charge density factor ρ_0 . Observe that for a monoenergetic beam

$$\mathbf{j}(\mathbf{x}) = v\rho(\mathbf{x}) = \rho_0 v\rho_p(\mathbf{x}) = \rho_0 \mathbf{j}_p(\mathbf{x}) \quad (67)$$

where $\rho_p(\mathbf{x})$ and $\mathbf{j}_p(\mathbf{x})$ are profile functions for the charge and current densities respectively. Hence if the total current is I , then

$$I = \int_S \mathbf{j} \cdot d\mathbf{S} = \rho_0 \int \mathbf{j}_p \cdot d\mathbf{S} \quad (68)$$

where ρ_0 is defined in terms of beam area A_b and particle speed v , via

$$I = \rho_0 v A_b \quad (69)$$

The speed v is (from Eqs (63) and (65)) given in terms of the dimensionless momentum p' , by

$$v^2 = \frac{p'^2 c^2}{1 + p'^2} \quad (70)$$

Hence

$$N_{in} = \frac{\rho_0 j_p A_s}{N_{ps} q_s |e|} \quad (71)$$

where A_s is the area of the element surface contributing to S . Note that $\mathbf{j}_p(\mathbf{x})$ has to be normalised so that

$$\int \mathbf{j}_p \cdot d\mathbf{S} = A_b v \quad (72)$$

Converting to dimensionless units gives

$$N_{in} = \rho_0 j_p A_s \quad (73)$$

where

$$\rho_0 = \left[\frac{I}{A_b v} \right] \frac{L^3}{N_{ps} q_s |e|} \quad (74)$$

The dimensionless momentum follows from Eq (65), where in terms of pre-defined quantities, dimensionless voltage

$$V' = \left[\frac{qV}{m_0 c^2} \right] = \frac{e}{m_e} \frac{m_{Es}}{N_{ps}} [V] \quad (75)$$

where m_{Es} is entry MRATPS of the array SPATR.

Additional refinements allowed by the software include uniform beam rotation in cylindrical geometry about the axis ($r = 0$), and it is anticipated that a finite beam temperature will be allowed.

One difficulty which arises in practice is to inject a beam into a geometry where the edges of elements do not conform to the edge of a beam, eg. the problem of injecting a circular beam in a cartesian geometry. This can be

Table 6: Entries in BCATR for beam injection.

Position	Entry
1	2
2	species no.
3	NSIG
4	(temperature)
5	M_{geo}
6	geometry parameters
$M_{geo} + 6$	I'
$N_I + M_{geo} + 6$	\mathbf{p}_T
$N_p + N_I + M_{geo} + 6$	\mathbf{x}_{SN}

overcome by passing the coordinates \mathbf{x}_{SN} (in the global system) of the nodes in the particle injection surface S , together with M_{geo} parameters describing the beam geometry. In the case of a circular beam geometry $M_{geo} = 2$ to store first the inner beam radius, then the outer one. This enables PIC3D to reject particles with initial positions that lie outside the beam cross-section.

The preprocessor also calculates I' , the number of particles to be emitted per element in each Δt , using Gaussian quadrature to evaluate the j integral, and the momentum triple \mathbf{p}_T corresponding the velocity \mathbf{v} . The coefficients are stored in array BCATR starting at location MPABCA (IPATCH) + 1, in the order shown in Table 6. NSIG is number defining the time dependence of I' ; N_I and N_p have the obvious meanings, and in particular for a three-dimensional geometry, $N_p = 3N_I$.

Fractional I' are allowed for in PIC3D by adding r , a random number on the unit interval to I' (corrected for signal strength and subcycling effects), then taking the integer part to give the number of particles to be emitted on the element. Initial positions are chosen randomly subject to the geometrical constraints, but every particle in the same element is assigned the same \mathbf{p}_T .

3.3.3 Patch Buffer Data

Table 7 summarises the quantities used in transferring information between uniblocks via the gluepatch buffers.

To transfer from block 1 to block 2, values of particle exit position \bar{x}_I^i , momentum triple \mathbf{p}_T (defined below) and the dimensionless time for which it still has to move Δt_r are copied from the particle arrays in records of length MPBPOS * INPOS + MPBMTM * INMOM + MPBATR to the output buffer GPATO. Usually MPBPOS = MPBMTM = MPBATR = 1. However, in calculations where an underlying cylindrical geometry is used, MPBPOS = 2, MPBMTM = 1 and MPBATR = 0 consistent with the patch buffer description

Table 7: Gluepatch exchange buffer data

Location	Value
MBTOGP	buffer to global patch table pointers
MGPTOB	global patch to buffer pointers
MPBEGI	patch start address in input buffer
MPLINI	patch lengths in input buffer
NPINBI	number of patches in input buffer
MPBEGO	patch start address in output buffers
MPLINO	patch lengths in output buffers
NPINB	number of patches in output buffer
NPADB	pointer to first free location in GPATO
LPHOC	particle buffer head of chains array
LPNXT	particle buffer link list array
GPATI	input buffer
GPATO	output buffer
GPATT	temporary buffer

tion in [15]. At the same time as the copies into buffer, information on the output record is accumulated in MPBEGO, MPLENO, NPINB, MBTOGP and MGPTOB. Patch information is then transferred to the input indexing arrays MPBEGI, MPLENI and the input data buffer GPATI, and thence to the target block. Any permutations and sign changes needed are performed on the patch buffer data in subroutine PCXFRM which is called by PARTIO. The pointer tables MBTOGP and MGPTOB are used to access global information on block, process, etc. Since particles can exit a block in random order through any face, they need to be sorted into patches, and it is for this that the linked list addressing quantities LPHOC and LPNXT are introduced. In addition, particles can cross more than one block boundary in a timestep, and this leads to the need for an extra buffer GPATT.

The need to introduce a momentum triple is now explained. Transformations consisting of permutations and sign changes are adequate for position values (except that care has to be taken so that $\bar{x}^i < n_i$ after the positional transformation). However, in general there is a more complicated transformation for p_i , because, supposing the normal coordinate to be $i = 3$ in each block, \mathbf{e}_3 is not always continuous across block boundaries. The vector $\hat{\mathbf{e}}^3$ is continuous, suggesting that sign changes and permutations of the mixed triple (p_1, p_2, p^3) might be adequate. In fact the momentum has to be represented in buffer as $\mathbf{p}_T = (p_1, p_2, \sqrt{g}p^3)$ (or appropriate perturbations depending on the normal direction), where \sqrt{g} is the volume element of the block which is left.

To see this, note that on the block boundary, continuity implies

$$\mathbf{p} = p_i \mathbf{e}^i = p'_i \mathbf{e}'^i \quad (76)$$

where undashed quantities belong to the block (1) which is left, and dashed ones to the block (2) which is entered. By construction \mathbf{e}_1 and \mathbf{e}_2 are the same for each block. Since for (ijk) permutations of (123), by definition

$$\mathbf{e}^i = \frac{\mathbf{e}_j \wedge \mathbf{e}_k}{\sqrt{g}} \quad (77)$$

and it follows that dotting Eq (76) with \mathbf{e}_1 and \mathbf{e}_2 respectively gives

$$p_1 = p'_1 \quad (78)$$

$$p_2 = p'_2 \quad (79)$$

Thus the tangential momentum is continuous. To treat the normal direction, note that, analogously to Eq (78),

$$\mathbf{p} = p^i \mathbf{e}_i = p'^i \mathbf{e}'_i \quad (80)$$

Dotting with \mathbf{e}^3 gives

$$p^3 = p'^3 \mathbf{e}'_3 \cdot \mathbf{e}^3 \quad (81)$$

and upon using Eq (77),

$$\sqrt{g}p^3 = \sqrt{g}' p'^3 \quad (82)$$

Hence the inter-block transfer is a three-stage process :

(i) Compute $\mathbf{p}_T = (\mathbf{p} \cdot \mathbf{e}_1, \mathbf{p} \cdot \mathbf{e}_2, \mathbf{p} \cdot \mathbf{e}^3 \sqrt{g})$ upon discovering that block 1 is to be left

(ii) Apply permutation and sign changes (due to the different orientations of two blocks' coordinate systems) to \mathbf{p}_T in the particle patch buffers

(iii) Compute $p'_i = (p_{T1}, p_{T2}, p'_3)$ on entry to block 2. The quantity p'_3 is given by :

$$p'_3 = \frac{p_{T3} - \sqrt{g'} \mathbf{e}^{3'} \cdot (\mathbf{p}_{T1} \mathbf{e}^{1'} + \mathbf{p}_{T2} \mathbf{e}^{2'})}{\sqrt{g'} \mathbf{e}^{3'} \cdot \mathbf{e}^{3'}} \quad (83)$$

which follows from the identity $p^3 = \sum_j g^{3j} p_j$

Note that if block 1 is orthogonal

$$p_{T3} = p_3^s \frac{e_{11} e_{22}}{e_{33}^s} \quad (84)$$

where $e_{(i)(i)} = \mathbf{e}_{(i)}^2$ and superscript s denotes values at the start of the time-step. If block 2 is orthogonal, Eq (83) simplifies to

$$p'_3 = \frac{e_{33}}{e_{11} e_{22}} p_{T3} \quad (85)$$

No other worthwhile simplifications seem possible.

3.4 Modifying Initial Conditions

Let us introduce the magnetic vector potential $\mathbf{A}(\mathbf{x}, t)$. For \mathbf{A} to represent an eigensolution of the discrete equations solved by PIC3D, it must satisfy

$$-\partial_t^2 \mathbf{A} = G^E \nabla_d \times G^H \nabla_d \times \mathbf{A} = \omega_d^2 \mathbf{A} \quad (86)$$

where ∂_t and ∇_d are the discrete representations of the time differentiation operator and the gradient operator respectively, G^E and G^H are the metric tensors and ω_d is the approximate eigenfrequency. The initial conditions employed by PIC3D use \mathbf{A}_{an} , the solution of the continuous version of Eq (86)

$$\nabla \times \nabla \times \mathbf{A} = \omega^2 \mathbf{A} \quad (87)$$

The two potentials will in general differ slightly, so

$$\mathbf{A} = \mathbf{A}_{an} + \epsilon \quad (88)$$

Writing

$$L_H = G^E \nabla_d \times G^H \nabla_d \times \quad (89)$$

it follows that

$$(L_H - \omega_d^2) \mathbf{A}_{an} = (\omega_d^2 - L_H) \epsilon \quad (90)$$

thus if

$$L_H \epsilon = \lambda^2 \epsilon \quad (91)$$

then

$$\epsilon = \frac{(L_H - \omega_d^2) \mathbf{A}_{an}}{(\omega_d^2 - \lambda^2)} \quad (92)$$

It follows that the field \mathbf{A} should be a more accurate solution of the discrete Helmholtz equation, where

$$\mathbf{A} = \frac{(L_H - \lambda^2)\mathbf{A}_{an}}{(\omega_d^2 - \lambda^2)} \quad (93)$$

The significance of the L_H operator is that its effect is equivalent to one pass through the main time step loop of PIC3D, thus the amount of extra coding needed is kept to a minimum. In Eq (93) the quantity λ is estimated using

$$\lambda^2 = \frac{\langle L_H \epsilon, L_H \epsilon \rangle}{\langle \epsilon, \epsilon \rangle} \quad (94)$$

where \langle, \rangle denotes an inner product on the space of vector functions. There are a number of ways to estimate ω_d , of which $\omega_d \approx 0$ is employed for the magnetostatic problem considered below. Another possibility is

$$\omega^2 = \frac{\langle L_H \mathbf{A}_{an}, \mathbf{A}_{an} \rangle}{\langle \mathbf{A}_{an}, \mathbf{A}_{an} \rangle} \quad (95)$$

In practice, the inner product is replaced by the sum of an element-wise, component-by-component product of vectors.

4 Postprocessors MPICTIM and DISPAN

There are now two main codes for postprocessing PIC3D output, MPICTIM which was described in Ref [16] and DISPAN for dispersion analysis. The enhancements to MPICTIM should be apparent to the user, and easily assimilated by anyone familiar with Ref [16]. The dispersion analysis postprocessor DISPAN allows the interactive interrogation of the ".lin" datasets generated by PIC3D.

The dispersive properties of a microwave structure are measured by setting up a periodic simulation containing an integral number of cavities, and collecting data as a function of space (axial position) and time. The data collected and its sampling frequency is user selected in the same manner as for the time series data sets used by MPICTIM. The user then follows the DISPAN operating sequence to extract frequency and wavenumber information from datasets generated by PIC3D.

DISPAN takes the user through the following sequence of operations:

1. file selection
2. domain selection
3. field selection
4. spatial fourier transformation
5. selection of time interval
6. time FFT

7. structure factor display
8. frequency spectra at given k
9. further selection options

The input to the dispersion analysis program DISPAN is a two dimensional array of numbers $A(p, q) \equiv A(z_p, t_q)$; $p = 1 \dots NZ$, $q = 1 \dots NT$ and arrays of position coordinates $\{z_p : p \in [1, NZ]\}$, and time coordinates $\{t_q : q \in [1, NT]\}$. In the present version of the software, it is assumed that

1. $z_p = z_o + (p - 1)\Delta z$, where Δz is the spatial mesh spacing, and there is period length $L = NZ \times \Delta z$, i.e.

$$A(z + nL, t) = A(z); n \text{ integer.}$$

2. $NZ = 2^r 3^s 5^t$ where r, s and t are integers ≥ 0 .

To reduce the length of computations required to construct the structure factor $S(\beta, f)$, we assume that A is periodic in time rather than use the usual cosine-bell envelope over many cycles of oscillation. Interactive selection of the temporal periodicity allows the period to be chosen to fit the period of one of the longer wavelength modes (eg the mode with $\beta = \pm 2\pi/L$); this allows reasonably accurate estimates of the eigenfrequencies $f = f(\beta)$ of the long wavelength modes to be obtained from just one or two periods of oscillation.

The stages of analysis are:

1. Spatial fourier transformation.

$$\hat{A}(k, q) = \sum_{p=0}^{NZ-1} A(p, q) \exp \left[\frac{i2\pi kp}{NZ} \right] \quad (96)$$

2. The period selection process selects a subset of data values $\{t_q, \hat{A}(k, q) : q \in [q_L, q_U]\}$ taken to be an integral number of oscillation periods.
3. Cubic spline fitting to $\{t_q, \hat{A}(k, q)\}$.
4. Interpolation to a uniform net of $NTNEW = 2^Q$ modes, where $2^{Q-1} < q_U - q_L + 1 < 2^Q$ and the time interval between nodes $DTNEW = (t_{q_U} - t_{q_L})/NTNEW$. The results is the set of values $\hat{A}^\dagger(k, q)$
5. Temporal fourier transformation

$$\tilde{A}(k, L) = \sum_{q=0}^{NTNEW-1} \hat{A}^\dagger(k, q) \exp \left[\frac{i2\pi lq}{NTNEW} \right] \quad (97)$$

6. Conversion to amplitude-phase. The FFT software used transforms N real data values to $N/2 + 1$ real and imaginary part values for its hermitian transform. The doubly transformed data $\tilde{A}(k, l)$ is stored as the sum of four numbers for $k \in [0, NZ/2]$, $l \in [0, NTNEW/2]$:

$$\tilde{A}(k, e) = \tilde{A}_{ee} - \tilde{A}_{oo} + i(\tilde{A}_{eo} + \tilde{A}_{oe}) \quad (98)$$

where subscripts 'e' and 'o' denote even and odd in the indices i.e.

$$\begin{aligned} \tilde{A}_{eo}(k, l) &= \tilde{A}_{eo}(-k, l) \\ &= -\tilde{A}_{eo}(k, -l) \\ &= -\tilde{A}_{eo}(-k, -l) \end{aligned} \quad (99)$$

and so forth. Amplitudes S and phases ϕ for $NZ/2 < k < NZ/2$, $0 < l < NTNEW/2$ are given respectively by

$$S = |\tilde{A}| = \sqrt{(\tilde{A}_{ee} - \tilde{A}_{oo})^2 + (\tilde{A}_{eo} + \tilde{A}_{oe})^2} \quad (100)$$

$$\phi = \arctan \left(\frac{\tilde{A}_{eo} + \tilde{A}_{oe}}{\tilde{A}_{ee} - \tilde{A}_{oo}} \right) \quad (101)$$

Amplitude $S(\beta, f) \equiv S(k, l)$, where frequency f and wavenumber β are related to l and k by

$$f = \frac{\omega}{2\pi} = \frac{l}{NTNEW\Delta t}, \quad \beta = \frac{2\pi k}{NZ\Delta z} \quad (102)$$

The bidirectional spectrum gives contours of S in (k, f) space, or sections of S versus f for given values of k . The ratio of $S(-\beta, f)$ and $S(\beta, f)$ give the ratio of amplitudes of forward and backward waves; if $S(-\beta, f) = 0$, then the wave is purely left going $\sim \exp[i(\omega t + \beta z)]$ and if $S(\beta, f) = 0$, it is purely right going $\sim \exp[i(\omega t - \beta z)]$. For an example of a plot of S , see Fig 6 in Section 5.

The omnidirectional spectrum gives the amplitude as a function of β and f irrespective of direction:

$$S' = \sqrt{\tilde{A}_{ee}^2 + \tilde{A}_{oo}^2 + \tilde{A}_{eo}^2 + \tilde{A}_{oe}^2} \quad (103)$$

5 Sample Calculations

This Section is inevitably something of a picture show. Each of the test problems defined by the PEGGIE ".uif"s in Appendix A is described, and a sample of the consequent output from PIC3D, visualised using MPICTIM unless otherwise stated, is presented. Finally the simulation of a helix TWT is described.

5.1 Connectivity Test `une51`

To test the treatment of inter-block connectivities both in PEGGIE and PIC3D, a grid was constructed using eight one metre cubes that are joined (full-face) in every possible way. The result of attempting to make every block have a mesh $3 \times 4 \times 5$ is drawn in Fig 1. Note that the geometry is periodic, thus which of the images of a block appears depends on arbitrary details of the PEGGIE connectivity algorithm. However PEGGIE has successfully produced a consistent meshing, and the corresponding dataset `une51.dat` ready for input to PIC3D.

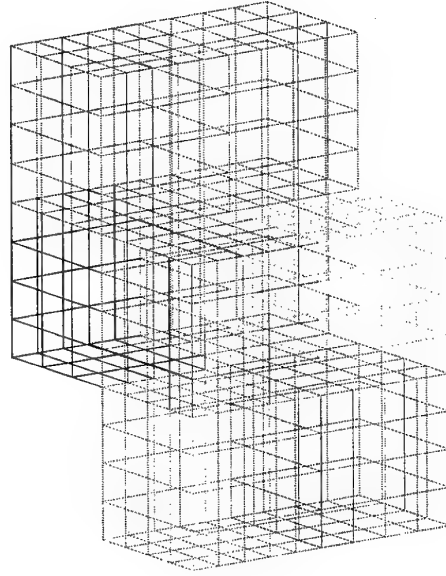


Figure 1: The meshing produced for the connectivity test `une51`. Each block is a metre cube.

The input dataset is designed to excite a TE_{nmp} mode with $n = m = p = 1$ of wavelength 2 metres in each coordinate direction. Such a mode, in a cavity of dimensions $a \times b \times d$, oscillates with a frequency of

$$f = \frac{c}{2} \sqrt{\left(\frac{n}{a}\right)^2 + \left(\frac{m}{b}\right)^2 + \left(\frac{p}{d}\right)^2} \text{ Hz} \quad (104)$$

where c is the speed of light. Hence at any given station for `une51` parameters, a regular oscillation at $f = 259.8$ MHz, corresponding to a period of 3.85ns, is to be expected. Figure 2, a run of 100 steps with timestep $\Delta t = 3.849 \times 10^{-10}$ s, bears out this prediction. The amplitude of the electric field reflects a dimensionless value of $E_0 = \mathcal{O}(1)$, since this translates to a dimensional value of $E_0 = 2m_e c / (|e| \Delta t) \approx 1.8 \times 10^6 / h$ for unit Courant number, where h is a measure of mesh spacing in metres.

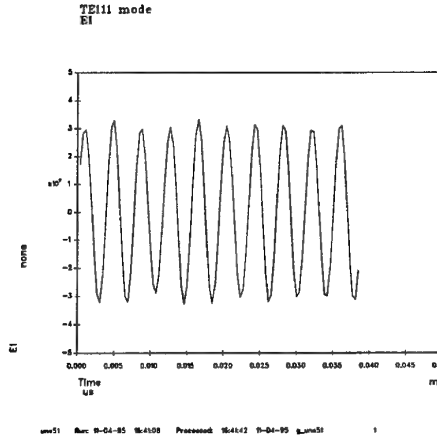


Figure 2: Point values of electric field component E_x in Vm^{-1} plotted as a function of time t in μs for the PIC3D run une51.

5.2 Nonorthogonal Test tp06

To test electromagnetic phenomena in a nonorthogonal geometry, a grid was constructed, consisting of a single parallelepiped with periodic boundary conditions in each direction. The parallelepiped took the form of a cylinder of height 2 m, which in cross-section is a parallelogram of sides 2 and $2\sqrt{2}$ metres and angle of 45° . After some minor changes, PEGGIE successfully produced a dataset containing the correct patch information.

The initial condition was a TE_{111} mode in the unit cube, which provides a spatially triply periodic solution to Maxwell's equation in the cube of side 2. The idea behind the choice of geometry of the parallelepiped is to provide an alternate "tiling" of three-space. Thus this meshing is simply a strange way of representing the unit cell of a triply periodic geometry.

Figure 3 shows that at the station in the centre of the parallelepiped, the electric field exhibits beats. The run involved 200 timesteps, the mesh used was $10 \times 10 \times 10$ and the timestep $\Delta t = 2.722 \times 10^{-10}s$.

Beats are to be expected on the basis of the following argument. The TE_{111} mode in cartesian coordinates corresponds to a magnetic vector potential

$$\mathbf{A} = (\cos \pi x \sin \pi y \sin \pi z, -\sin \pi x \cos \pi y \sin \pi z, 0) \quad (105)$$

The nonorthogonal geometry defined by tp06 has basis vectors given by

$$\mathbf{e}_1 = (b, 0, 0), \mathbf{e}_2 = (b, b, 0), \mathbf{e}_3 = (0, 0, b) \quad (106)$$

where b is a constant, the exact of value of which need not concern us. Hence the components of \mathbf{A} in this system are $A_j = \mathbf{A} \cdot \mathbf{e}_j$, or

$$A_1 = bA_x, A_2 = b(A_x + A_y), A_3 = bA_z \quad (107)$$

implying in the appropriately normalised nonorthogonal (x^1, x^2, x^3) system that

$$A_1 \propto \cos(x^1 + x^2) \sin x^2 \cos x^3, A_2 \propto -\sin x^1 \sin x^3, A_3 = 0 \quad (108)$$

Thus the A_j are a mixture of different spatial dependences. Analytically these must have same frequencies, since the spatially triply periodic extension of the TE_{111} mode has a unique frequency, however eccentrically the unit cube is meshed. However, discretisation effects will mean that the different spatial dependences correspond to slightly different propagation frequencies, that is to say, beats. To check further, the dispersion relation in Ref [11] was evaluated for modes with wave-vectors $\mathbf{m} = (1, 0, 1)$ and $(1, 2, 1)$ in the curvilinear system, i.e. we set $\bar{k}_j = m_j \pi / N_j$ where N_j is the number of elements in the j direction. The predicted frequencies are $f_1 = 269$ MHz and $f_2 = 313$ MHz respectively, corresponding to a mean frequency of 291 MHz (period 3.44 ns) and a beat frequency of 22 MHz (period 45 ns). These agree satisfactorily with the computed values of 3.3 ns and 50 ns respectively.

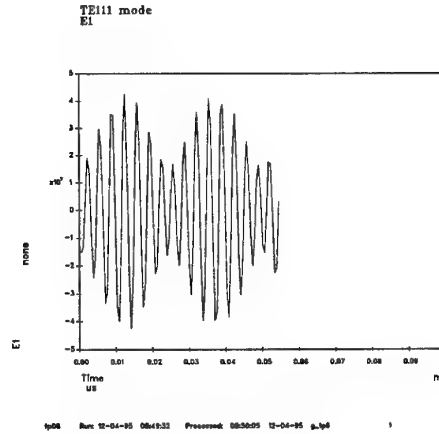


Figure 3: Point values of electric field component E_x in Vm^{-1} plotted as a function of time t in μs for the PIC3D run tp06.

5.3 Oscillations on Axis cy107

The dataset **cy107** was devised to investigate the appearance of spurious oscillations localised near the axis of a cylindrical device, when non-axisymmetric TE_{111} and TM_{111} modes are used as initial conditions. To reduce the problem to essentials, **cy107** involves a single block representing a cylindrical cavity in perfectly conducting material which is 1 mm in radius and 2 mm long. The magnetic field is initialised as a constant vector pointing in the transverse x -direction: in the absence of electric field, this is a steady state solution of Maxwell's equations.

Figure 4 is a superposition of results from two runs, without and with the correction to the initial conditions discussed in Section 3. (The amplitude in the

plots is essentially arbitrary since the θ and r components of \mathbf{H} become multiply valued at the origin.) The former curve possesses the spurious oscillation, the latter shows it at much reduced amplitude. In each case a mesh $8 \times 8 \times 16$ was used with a timestep $\Delta t = 9.447 \times 10^{-12}$ s. The conclusion to be drawn is that the spurious oscillations are simply a consequence of the failure of the initial, analytic fields to solve the relevant discrete Helmholtz problem for a cylindrical cavity. Hence for many purposes they may be neglected.

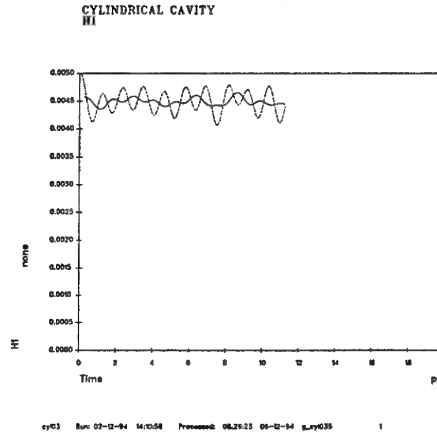


Figure 4: The value near the origin of the magnetic field component H_r in arbitrary units plotted as a function of time t in ps for the PIC3D run cy107.

5.4 Surface Conditions sbc16z

The problem was to test the surface conditions. A TM_{111} mode has a frequency of 25.98 GHz in a cubical cavity of side 10cm. The corresponding relative impedance is given by

$$\frac{1}{Z_r} = \sqrt{1 + \left(\frac{nd}{ap}\right)^2 + \left(\frac{md}{bp}\right)^2} \quad (109)$$

for a general TM_{nmp} mode in a cavity of dimensions $a \times b \times d$. Thus setting $n = m = p = 1$, $a = b = d = 0.01$, gives $Z_r = 0.5769$, corresponding to $Z_s = 217.51\Omega$. A TM_{11} mode of axial wavelength $2d$ excited in such a geometry with the corresponding impedance at entry and exit planes should travel as though in an infinite waveguide.

The coefficients in the file **sbc16z.dat** generated by PEGGIE were checked by hand, then the dataset was used to control a PIC3D run. Fig 5 shows that, after an initial transient, point values of E_y at different z , co-ordinate along the waveguide, fit the pattern of a mode propagating along an infinite waveguide. Details of the computation are that it used an $6 \times 8 \times 10$ mesh and timestep $\Delta t = 1.925 \times 10^{-12}$ s corresponding to unit Courant number.

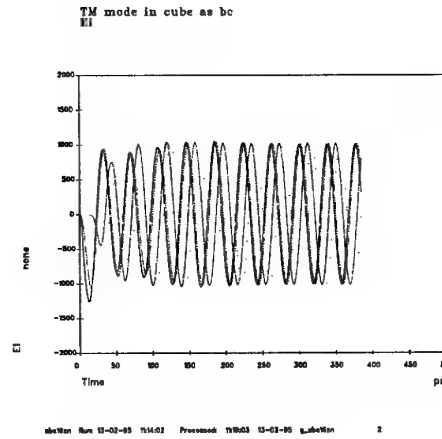


Figure 5: Point values of E_x at different z , plotted as functions of time in ps , for the model rectangular waveguide in run `sbc16z`.

5.5 MILO Cavity Dispersion `mlp903`

Here the geometry is one segment of an annular Magnetically Insulated Line Oscillator (MILO), ie two concentric annular cavities, the outer of which has smaller axial extent than the inner. Periodic boundary conditions are imposed in the axial direction, to represent a device of infinite length for which analytic field solutions are available. The dimensions are (outer cavity in parenthesis), inner radius = 3.75cm (7.5cm), outer radius = 7.5cm (15cm), axial extent = 5cm (4cm). Each annular cavity is, for meshing purposes, divided in angle into three equal segments, each of which has a $8 \times 4 \times 20$ ($8 \times 4 \times 16$) mesh, implying 8 elements in the radial direction, 4 in polar angle θ and 20 (16) elements in z along the axis.

In order that a range of wavelengths is excited large enough to compare with analytic results, the dataset `mlp903.uif` represents a nine-cavity MILO. The ability to define a line by extension and inheritance is particularly useful here, where a line parallel to the z -axis crosses nine blocks: the resulting plots were most helpful in checking the initialisation procedure. The analytic comparison involves purely electromagnetic waves in vacuo, but the method of excitation involves switching on, for the first timestep only, a current directed along z in one MILO segment. The run `mlp903` had $\Delta t = 4.750 \times 10^{-12}s$ and continued for 5ns, during which time data were output to a `.lin` file. Subsequent analysis by DISPAN produced the contour plot $S(k, l)$ drawn in Fig 6. Overlaying the contours are dispersion curves obtained analytically, assuming periodicity and the narrow cavity approximation. The agreement is excellent for the curve below the first band gap. The anomalies above may be explained by the comparatively short time series.

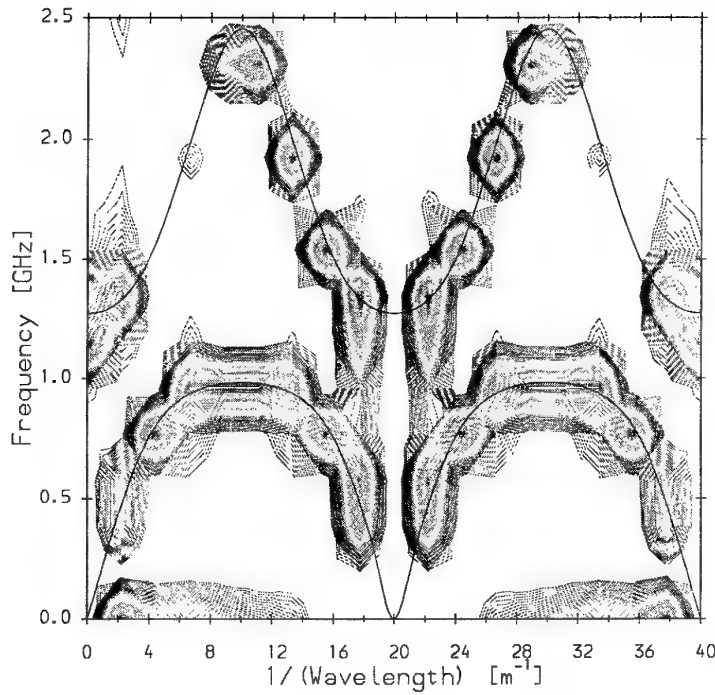


Figure 6: Dispersion test for periodic MILO geometry. The contours are of amplitude normalised structure factor $S(k, l)$ for E_r obtained from the run mlp903, whereas the graph of dispersion curves is obtained analytically. Note the origin of wavenumber k for the contours is at the centre of the plot, not at the left as implied by the labelling.

5.6 Particle Orbit Tests porb21, porb35 and porb40

This Section demonstrates the ability of PIC3D to model particle orbits in a complicated geometry, essentially that of **une51** which has all possible inter-block connectivities. For convenience, each cube is now taken to have side 1 cm and a $4 \times 4 \times 4$ meshing. The orbit tests involve runs of 400 timesteps with $\Delta t = 4.811 \times 10^{-12}$ s. No electric field is present, and the magnetic field has only one component with a uniform value of 0.054 T, chosen so that the electron gyro-radius is $\sqrt{5}/2 \approx 1.118$ cm for an orbital speed of $c/3 \approx 1 \times 10^8$ m/s.

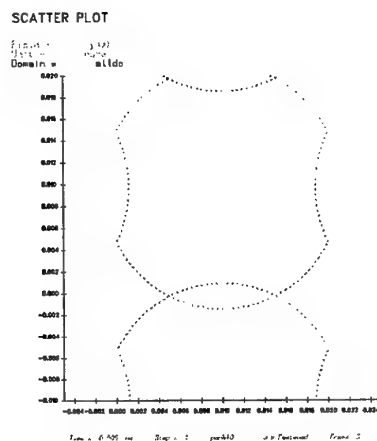


Figure 7: Plot of particle position projected onto the plane normal to B (yz -plane), in the cluster of eight cubes for each of the first 400 timesteps of run **porb40**.

It will be noted that the diameter of the gyro-orbit exceeds the 2 cm dimension of the cube cluster, so that in the orbital plane each inter-block connection is sampled. The result in plotting space is an indented circular orbit, shown in Fig 7, which exhibits the ability of PIC3D to generate particle scatter plots directly (such files are suffixed **.gst**, and the figure represents a collage of all the frames in such a file). The basic orbit shown is clockwise, but the indents appear to be traversed anticlockwise. Imposing an initial velocity with an extra component, parallel to the applied field, ensures that every normal face is sampled, and hence in Fig 7 that one and a half copies of the basic orbital pattern appear.

The run **porb40** is for the case where only $B_x \neq 0$, and the centring of the orbit implies that a component of velocity is zero at some boundaries. Moving the orbit centre by an amount of order 0.1 cm leads to a more complete test, exemplified by Fig 8 for the case where only $B_z \neq 0$ and Fig 9 for $B_y \neq 0$. In each case the orbital period is $P = 7.01 \times 10^{-10}$ s, so that 2.75 cycles are covered in a run. It is also arranged so that the particle travels about 2.9 cm parallel to B . Exact details may be obtained from the ".uif" listed in Appendix A. We see that in all cases the orbits are closed and have the expected radius.

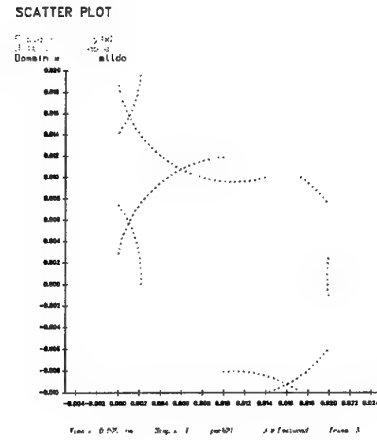


Figure 8: Plot of particle position projected onto the plane normal to B (xy -plane), in the cluster of eight cubes for each of the first 400 timesteps of run porb21.

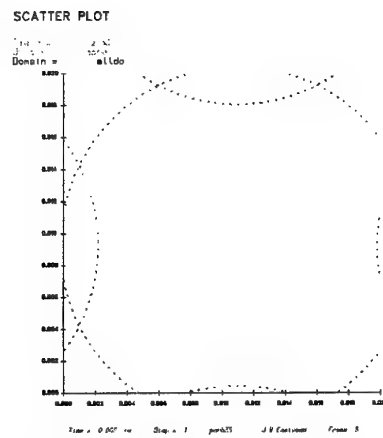


Figure 9: Plot of particle position projected onto the plane normal to B (xz -plane), in the cluster of eight cubes for each of the first 400 timesteps of run porb35.

5.7 Orbit Tests `porg60h` and `porg70` in General Geometry

There are essentially two new geometry types to test, since orthogonal coordinate systems were covered in the previous Section. The first is the generalised cylindrical geometry shown in Fig 10. The central region is gridded by three right cylinders which are parallelograms in cross-section. Between the parallelograms and the bounding circle of radius 1 mm are six blocks where the e_i lying in the cross-section plane have spatial dependence. The total number of distinct nodes in the cross-section is 132 and there are 6 elements in the axial direction. The magnetic field is 2.16 T, chosen so that a particle with an initial speed of 10^8 m/s has a gyro-orbit of diameter 0.559 mm. The subcycling parameter $N_s = 10$, thus for a Courant limited time-step of 2.083×10^{-13} s, there are 8.4 particle moves per orbit.

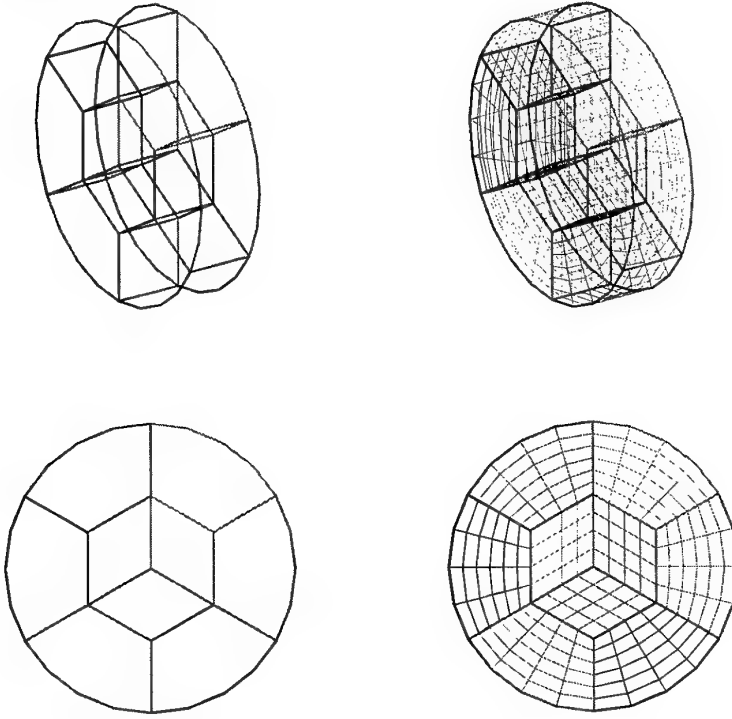


Figure 10: The meshing for the particle orbit test `porg60h`.

As expected, particle orbits confined to the parallelogram meshed regions are closed and repeat exactly. Such a test is a severe examination not only of particle motion within blocks, but also of the transformations applied at block boundaries. Unfortunately, the situation for orbits that cross the outer blocks is less clear cut, Figure 11 was produced from a `.pts` output file using GNUPLOT. It shows two numerically computed orbits for the run `porg60h`, the upper starting at a radius $r = 0.2795$ mm, on the y axis with a velocity of 10^8 m/s directed only in x , such as to cause outwards travel. The lower starts at $r = 0.75$ mm on the y axis with a similar velocity, but directed so as to

travel inwards. Although the upper orbit is not closed, throughout the run of 400 time-steps or nearly 50 orbits, it remains a satisfactory approximation to the expected motion. The lower orbit clearly precesses by a significant amount. Other similar runs show no obvious pattern in the way some orbits precess, whilst others remain relatively well behaved.

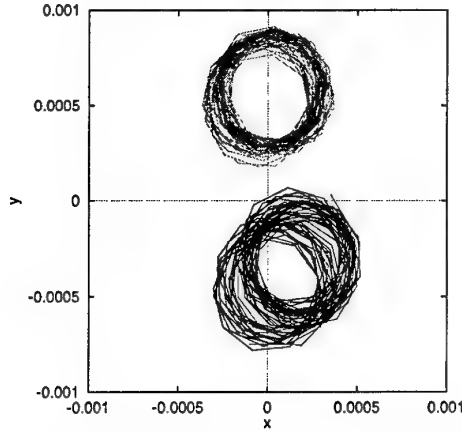


Figure 11: Plot of two particle orbits each projected onto the plane normal to B (xy -plane), in the threefold symmetric meshing of the cylinder used by the run `porg60h`.

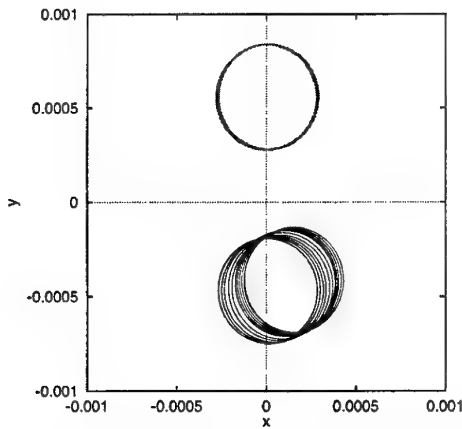


Figure 12: Plot of two particle orbits each projected onto the plane normal to B (xy -plane), in the orthogonal meshing of the cylinder used by the run `porb96`.

Shown in Fig 12 are results for run `porb96` which employs an orthogonal cylindrical geometry with a 12×12 mesh in cross-section. Here $N_s = 5$ and because of the smaller mesh size near the origin, this actually corresponds to 84 steps per orbit, of which approximately 12 are completed in 1000 steps. The random appearance of precession here cannot be attributed to discontinuities in the \mathbf{e}^i (since they are continuous), or to taking too coarse a time-step and applying ten corrector steps instead of one has no effect either. It seems that

it is primarily a consequence of the piecewise constant approximation used for the (x, y) dependence of b^i , which results in an effective $B_z \propto r^{-1}$ over a cell radius. Similar spurious spatial dependences within a cell are expected for the nonorthogonal geometry, and presumably lead to the observed precession.

The last orbit test concerns a completely general geometry, that of the parallelepiped for which the generating vectors are given by

$$e_{ij} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 2 & 2 & 1 \end{pmatrix} \quad (110)$$

where the units are millimetres. The total volume occupied is one cubic millimetre, and for orbit testing it is convenient to impose periodic boundary conditions.

The run `porg70` used a $10 \times 10 \times 10$ mesh and a time-step of 3.514×10^{-14} s, so that $N_s = 10$ implies 50 points in a gyro-orbit. Taking periodicity into account, Fig 13 shows the expected closed orbits. We conclude that PIC3D can accurately simulate particle motion in complex, non-orthogonal geometries.

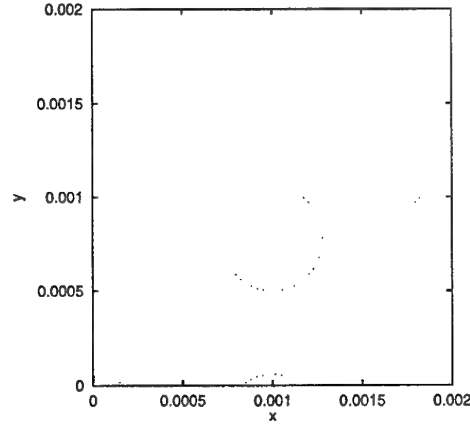


Figure 13: Plot of particle position projected onto the plane normal to B (xy -plane), in the general parallelepipedal geometry for each of the first 400 timesteps of run `porg70`.

5.8 Particle Emission m1076

The geometry is an annular Magnetically Insulated Transmission Line (MITL), sketched in Fig 14. The electromagnetic boundary conditions represent perfect conductors on A, B, C, D and E, with particle emission on D only, and F is a segment of the polar axis. The TEM mode with an electric potential amplitude of 500 kV is imposed at A, with field directed to pull electrons from D. The number of charges per superparticle is 5×10^9 . The main annular grid is $15 \times 12 \times 80$ (composed of three 120° blocks), $\Delta t = 1.247 \times 10^{-12}$ s and no subcycling was used. Fig 15 plots particle radius r against z . The distribution

of particle positions agrees qualitatively with that found in the corresponding two-dimensional test case, exhibiting magnetic insulation.

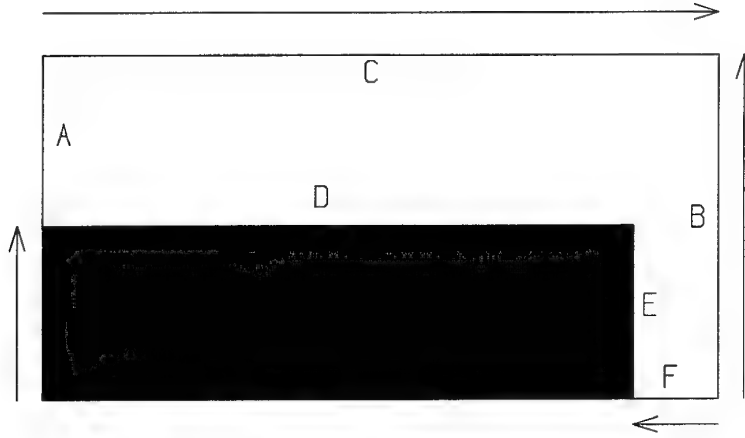


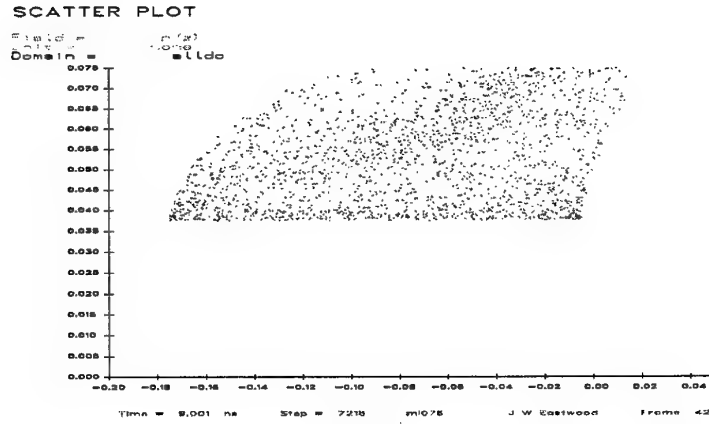
Figure 14: The cross-section of the annular MITL used for the test calculation of particle emission. The device is axisymmetric about the line containing F, and the dimensions (in cm) are $A = E = 3.75$, $B = 7.5$, $C = 20$ and $F = 1.5$.

5.9 Particle Beam Injection bean32

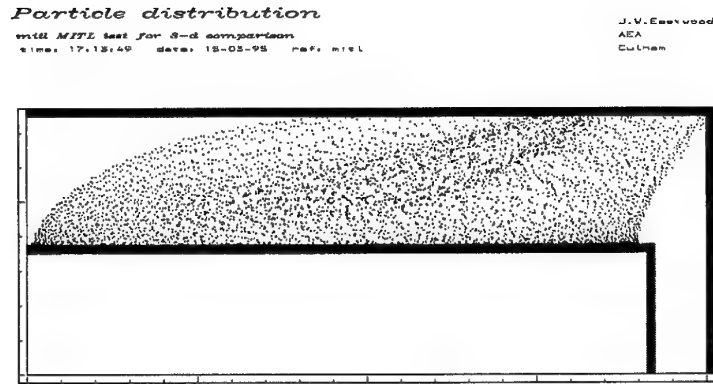
The geometry is a circular cylinder, but meshed in cross-section as shown in Fig 16. The cylinder radius is 1 mm, and the central cartesian block is chosen to have a maximum radial extent of 0.7 mm, ie. a side of length $0.7\sqrt{2}$ mm. The central square has 20×20 elements in cross-section, with sides of length approximately 0.05 mm. The device has a length of 16 mm in the axial (z -) direction, split up into four equal sections, so that there are 20 blocks altogether. The mesh spacing in z is 0.1 mm, and the timestep at unit Courant number is 6.055×10^{-13} s. The cylinder is taken to have perfectly conducting walls, except for the flat wall at extreme z which is chosen to be resistive with a surface impedance equal to that of free space.

A beam of electrons of circular cross-section with radius 0.3 mm is injected at $z = 0$. The beam voltage is 7.085 kV, and the total (absolute) current is ramped linearly up to 0.21 A, over an interval of 100 timesteps so as to reduce transient effects. The number of charges per superparticle is 6000, so that at peak power, an average of 0.583 particles are emitted per element in time Δt_p . The subcycling parameter $N_s = 5$.

Fig 17 plots particle radius r against z after 1000 timesteps. The distribution of particle positions agrees quantitatively with that expected from the universal beam spreading curve[17, Chap. 3], for a laminar beam influenced only by its self field at small perveance (dimensionless $K = 5.27 \times 10^{-3}$ in this case). The discrepancy between the curve and the particle plots is accounted for by the failure of high order quadratures to treat accurately step function integrals, such as are needed for elements at the beam edge. The result is a reduction in



(a) PIC3D output



(b) 2-D axisymmetric

Figure 15: Scatter plot of particle radius versus axial coordinate for the annular MITL, at time $t = 9.0 \text{ ns}$ in the steady state regime. At bottom is a plot of the corresponding steady state computed using a 2-D axisymmetric code.

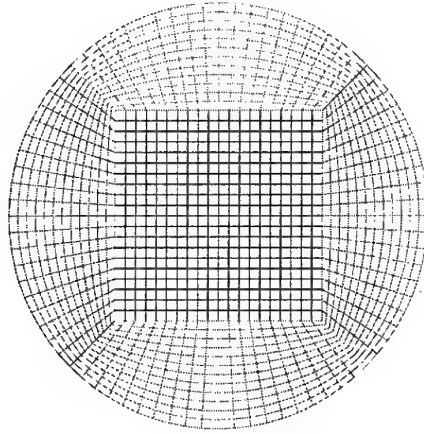


Figure 16: The meshing of the cross-section of the circular cylinder used for the test calculation of particle beam injection.

the effective beam current, hence less spreading of the beam.

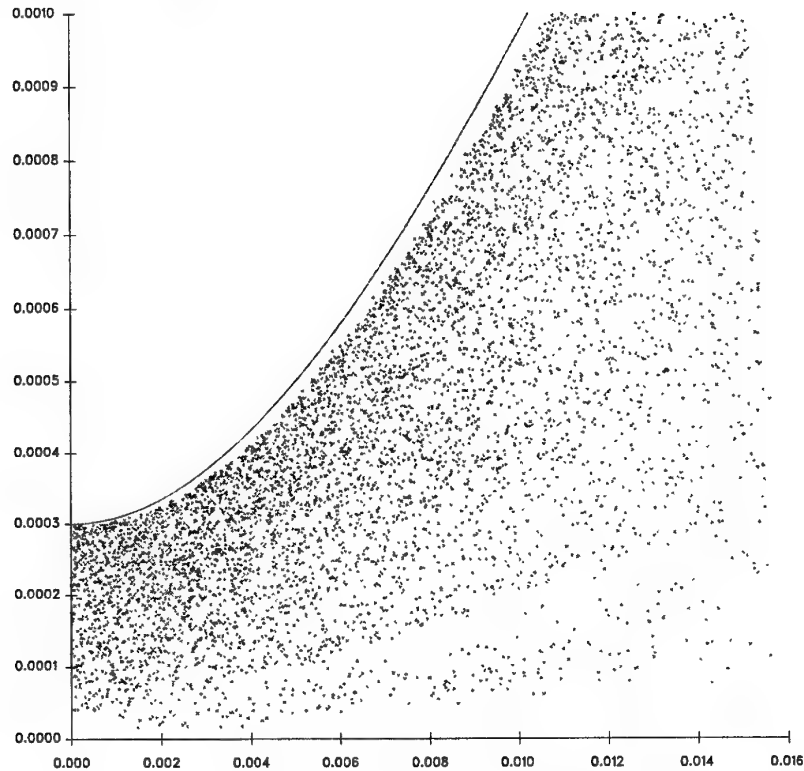
5.10 Four Cavity MILO m1407

The geometry is an annular Magnetically Insulated Line Oscillator (MILO), sketched at top left and bottom of Fig 18. Basically the MILO consists of the MITL with the addition of annular cavities on the outside that, in the example, extend to a radius of 7.5cm and which are 4cm in axial extent with a separation of 1cm. The same cavity segment was studied in the m1p903 calculations. The dataset m1407.uif represents a four-cavity MILO, with particle emission from all surfaces of the central, cylindrical cathode.

The plots of point values of field against time show oscillations with a period of about a nanosecond, see Fig 19, although the signals are rather noisy. The timestep used was $\Delta t = 2.338 \times 10^{-12}$ s (no subcycling) and the number of charges per superparticle is 6×10^9 . The run was continued to a time of 50 ns, when saturation of the signal was clearly apparent. The discrepancies in Fig 19 are accounted for by the different units employed, and that the PIC3D data was collected at a more interior point. We conclude that PIC3D is able to simulate electromagnetic particle effects in the complex geometries found in experiments.

SCATTER PLOT

Field = $r(z)$
 Unit = none
 Domain = all do



Time = 0.303 ns Step = 1000 bean32 J W Eastwood Frame 51

Figure 17: Scatter plot of particle radius versus axial coordinate for the run **bean32** with circular beam injection, at time $t = 0.3 \text{ ns}$ in the steady state regime. The line graph is a plot of the universal beam spreading curve at the corresponding parameters, computed by solving a simple ordinary differential equation.

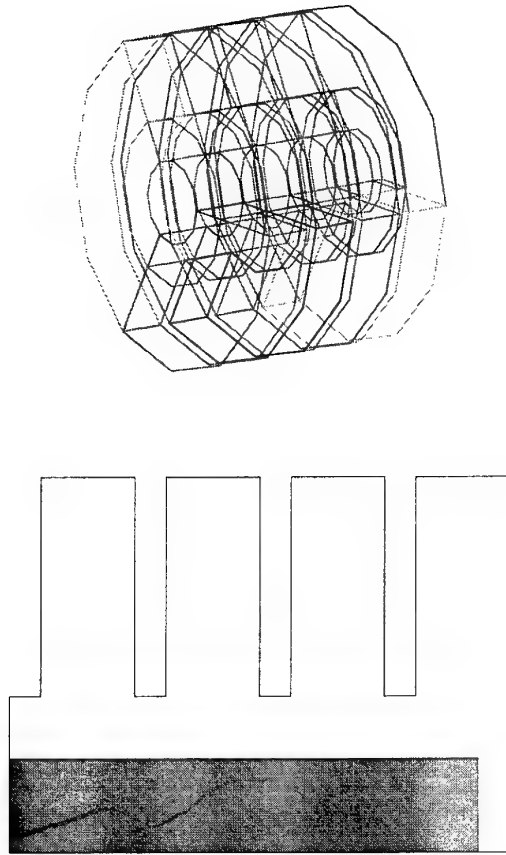
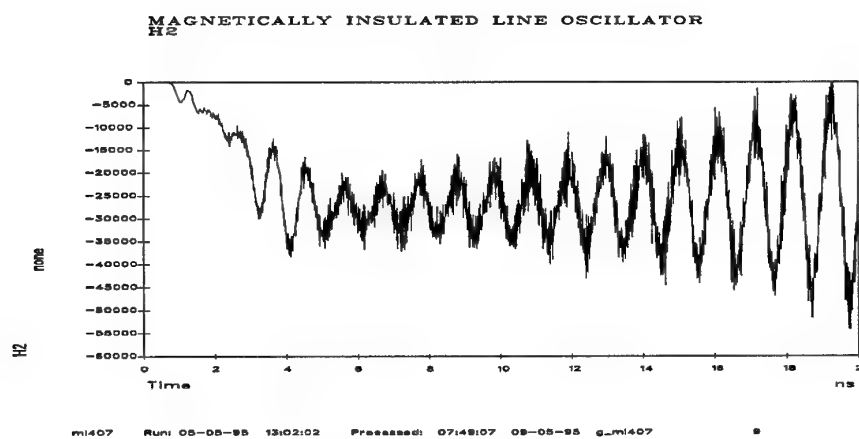
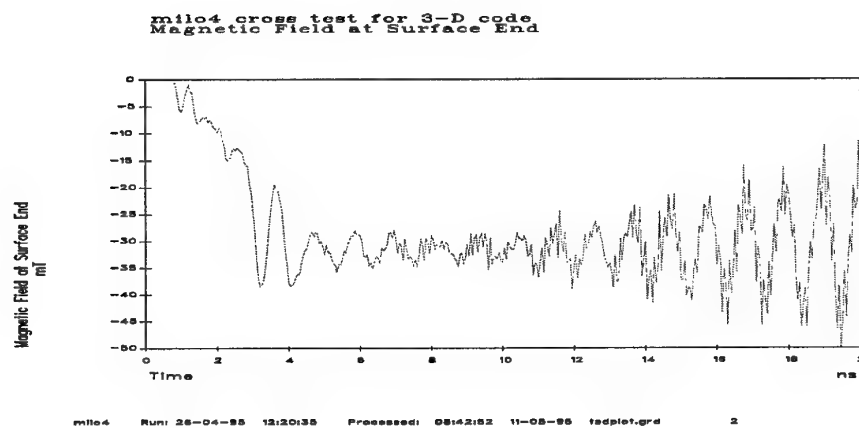


Figure 18: At the top is PEGGIE output showing a 4-cavity MILO and at the bottom is a sketch of the cross-section of the annular MILO: the device is axisymmetric about the centre-line, running left-to-right, of the shaded, cathode region.



(a) PIC3D output

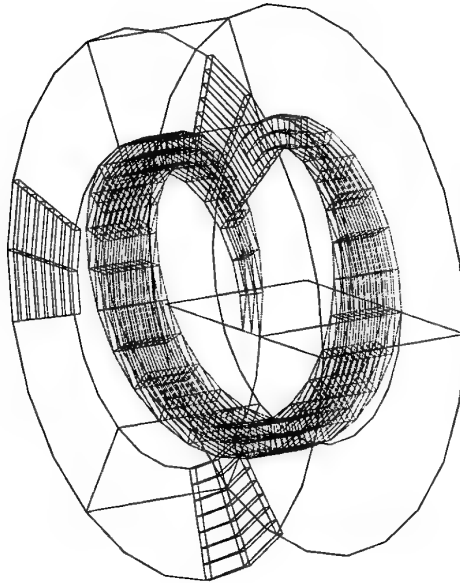


(b) 2-D axisymmetric

Figure 19: Point values of magnetic field component H_θ in $A - turn m^{-1}$ plotted as a function of time t in ns for the PIC3D run ml407. At bottom is the corresponding plot, only field given in units of mT , from a 2-D axisymmetric code.

5.11 Helix TWT pc11r01

Section 2



Date: 10-05-95 Time: 16:46:35 Run ID: pc11r01 Frame number: 6

Figure 20: Section of a helix TWT, showing the central helix and part of the tapered dielectric support, modelled by the PIC3D run pc11r01.

Unlike the previous tests, this was realised without use of PEGGIE, so that that only ".dat" file is listed in this Report, see Appendix B. The calculation simulates a loop of helix which is excited by 'pinging' the start of the first turn, in a manner similar to that employed by run mlp903. The geometry of one of the turns or sections of the TWT is shown in Fig 20. Other details (except dimensions which have been deleted to preserve proprietary information) of the TWT modelled may be inferred from the following log data:

General Run Data

=====

Run identification string: pc11r01

Remarks:

2 turn cold loopback test

Run time: 4.000E-10 s

Time series start time: 0.000E+00 s
 Time series end time: 1.000E-08 s
 Time series output interval: 4.000E-12 s
 Snapshot start time: 0.000E+00 s
 Snapshot end time: 1.000E-08 s
 Snapshot output interval: 2.000E-10 s
 Dispersion start time: 0.000E+00 s
 Dispersion end time: 1.000E-08 s
 Dispersion output interval: 4.000E-12 s

Section Data

=====

Number of Sections: 3

number	TYPE	LENGTH	SNAP	RECORD	ACTIVE	REFERENCE Z
----	----	-----	----	-----	-----	-----
1	HELIX	*****E-03	no	yes	yes	
2	HELIX	*****E-03	no	yes	yes	
3	LOOPBACK	0.000E+00	no	yes	yes	

First Active Section: 1
 Number of Active Sections: 3

number	Peak PPM (T)	PPM Period (m)	PPM Zero (m)	Uniform B (T)
----	-----	-----	-----	-----
1	0.000E+00	0.000E+00	0.000E+00	*****E-01
2	0.000E+00	0.000E+00	0.000E+00	*****E-01
3	0.000E+00	0.000E+00	0.000E+00	*****E-01

Nominal losses in dB/m:

number	Body Tube	Helix	Vane
----	-----	-----	----
1	0.000E+00	0.000E+00	0.000E+00
2	0.000E+00	0.000E+00	0.000E+00
3	0.000E+00	0.000E+00	0.000E+00

Section Number: 1
 Section Type: HELIX

Length: *****E-03 m
 Body tube radius: *****E-03 m
 Helix pitch: *****E-03 m
 Helix radius: *****E-03 m
 Helix width: *****E-04 m
 Helix thickness: *****E-04 m
 Ping helix: YES

Supports: 3
 Support angles: 6.000E+01 degrees
 Support angles: -6.000E+01 degrees
 Support angles: 1.800E+02 degrees
 Support Type: WEDGE
 Support perm.: *****E+00
 Support loss: *****E+00 dB/m
 Support heights: 0.000E+00 m
 Support heights: 0.000E+00 m
 Support chords: *****E-03 m
 Support chords: *****E-04 m
 Support chords: 0.000E+00 m
 Support chords: 0.000E+00 m
 Vanes: 3
 Vane angles: 0.000E+00 degrees
 Vane angles: 1.200E+02 degrees
 Vane angles: -1.200E+02 degrees
 Vane Type: NONE
 Vane loss: 0.000E+00 dB/m
 Vane heights: 0.000E+00 m
 Vane heights: 0.000E+00 m
 Vane chords: 0.000E+00 m
 Vane chords: 0.000E+00 m
 Vane chords: 0.000E+00 m
 Vane chords: 0.000E+00 m

Section Number: 2
 Section Type: HELIX

Length: *****E-03 m
 Body tube radius: *****E-03 m
 Helix pitch: *****E-03 m
 Helix radius: *****E-03 m
 Helix width: *****E-04 m
 Helix thickness: *****E-04 m
 Ping helix:
 Supports: 3
 Support angles: 6.000E+01 degrees
 Support angles: -6.000E+01 degrees
 Support angles: 1.800E+02 degrees
 Support Type: WEDGE
 Support perm.: *****E+00
 Support loss: *****E+00 dB/m
 Support heights: 0.000E+00 m
 Support heights: 0.000E+00 m
 Support chords: *****E-03 m
 Support chords: *****E-04 m
 Support chords: 0.000E+00 m
 Support chords: 0.000E+00 m
 Vanes: 3
 Vane angles: 0.000E+00 degrees
 Vane angles: 1.200E+02 degrees

Vane angles: -1.200E+02 degrees
 Vane Type: NONE
 Vane loss: 0.000E+00 dB/m
 Vane heights: 0.000E+00 m
 Vane heights: 0.000E+00 m
 Vane chords: 0.000E+00 m
 Vane chords: 0.000E+00 m
 Vane chords: 0.000E+00 m
 Vane chords: 0.000E+00 m

Section Number: 3
 Section Type: LOOPBACK

Twist: 0.000E+00 degrees
 Invert:

Beam Data
 =====

Device will be empty initially

Number of Intervals: 1

Radial Interval (m)	Radial Momenta (Mc)	Form
[0.000E+00 0.000E+00]	[0.000E+00 0.000E+00]	----
Radial Interval (m)	Azimuthal Momenta (Mc)	Form
[0.000E+00 0.000E+00]	[0.000E+00 0.000E+00]	----
Radial Interval (m)	Axial Momenta (Mc)	Form
[0.000E+00 0.000E+00]	[0.000E+00 0.000E+00]	----
Radial Interval (m)	Current (A/m**2)	Form
[0.000E+00 0.000E+00]	[0.000E+00 0.000E+00]	----
Radial Interval (m)	Particle Weight	Form
[0.000E+00 0.000E+00]	[0.000E+00 0.000E+00]	----

Current History Data
 =====

Number of Intervals: 1

Duration (s)	Current (A)
0.000E+00	[0.000E+00 0.000E+00]

Duration (s)	RF Current (A)
0.000E+00	[0.000E+00 0.000E+00]

Signal Data

=====

Number of Intervals: 0
 Zc(input): 0. ohms

Mesh (Nominal) Data

=====

Nominal current:	*****E-01 A
Nominal beam radius:	*****E-04 m
Nominal frequency:	5.000E+09 Hz
Nominal momentum (Mc):	*****E-01
Nominal helix pitch:	*****E-03 m
Nominal delta r:	*****E-04 m
Nominal delta z:	*****E-04 m
Nominal delta theta:	1.500E+01 degrees
Particles per element:	0.000E+00
Particles width:	*****E-04 m
Particles height:	*****E-04 m
Particles theta size:	0.000E+00 degrees

This example takes about 20 minutes to run on a Sun SPARCstation 20, and produces the following files:

pc11r01-01.gr1 which contains plots of fluxes (through planes perpendicular to the z axis) against distance along the z axis at the "snapshot" times. The quantities plotted are total displacement flux $\int \mathbf{D} \cdot d\mathbf{S}$, total magnetic flux $\int \mathbf{B} \cdot d\mathbf{S}$, and current $\int \mathbf{j} \cdot d\mathbf{S}$.

pc11r01-02.lin which is intended to be read by the dispersion analysis postprocessor DISPAN. It contains the voltage between the outer surface of the helix and the body tube at $\theta = 0$ as a function of axial distance, recorded at the "dispersion" times.

pc11r01-03.gr1 which contains plots of the field components E_r , E_θ , E_z , H_r , H_θ , H_z , on lines parallel to the z axis at three different radii - half an element away from the axis, approximately half way between the axis and the helix, and approximately half way between the helix and body tube. The plots are produced at the snapshot times.

pc11r01-04.tsd in the TSD format that can be read by MPICTIM. This

file contains global energy and particle diagnostics. It contains the total number of particles, the total electric energy, the total magnetic energy, the total electromagnetic energy and the total particle power. The data is collected at the "time series" times.

`pc11r01-05.gst` which normally contains particle scatter plots but only contains empty plots in this case as there are no particles.

`pc11r01-06.lin` which is intended to be read by the dispersion analysis postprocessor DISPAN. It contains plots of the axial electric field E_z on lines parallel to the z axis at two different radii – half an element away from the axis and approximately half way between the axis and the helix. The output is produced at the dispersion times.

`pc11r01-07.gr1` is not produced by this run because the snapshot (SNAP) variable is not set to 'YES' in any of the TWT sections. It would normally contain plots of field components against radius along the line $\theta = 0$ at the start of each section.

`pc11r01-08.gst` are not produced by this run because the SNAP variable is not set to 'YES' in any of the TWT sections. It normally contains particle scatter plots.

`pc11r01-09.tsd` in the TSD format. It contains values of the cylindrical polar components of \mathbf{E} and \mathbf{H} in the plane at the start of each TWT section, at points near the axis, half way between the axis and the helix and half way between the helix and the body tube. The data is collected at the time series times.

`pc11r01-10.tsd` in the TSD format. It contains the particle current fluxes through plane normal to the z -axis at the start of each TWT section. These values will all be zero in this case. The data is collected at the time series times.

`pc11r01.out` which contains run diagnostic data. It may contain error messages if a run stops prematurely. It also records the length of time the run took to complete, the name of the workstation the code was run on and the name and version of the operating system it was using.

Clearly the above files represent an enormous amount of information concerning the outcome of the TWT modelling process. The most important data are perhaps the tube dispersion properties. The first `.lin` file above may be used, in the same way as for the run `mlp903`, to produce a contour plot of structure factor, where the locations of the peaks give points on the dispersion curve. In this instance it proved necessary to make the run `pc11r02`, which has the number of helix turns increased to five, when a successful comparison was possible with (proprietary) experimental data, see Fig 21.

References

- [1] J W Eastwood, W Arter, N J Brealey and R W Hockney, *Body-fitted Electromagnetic PIC Software for Use on Parallel Computers*, Comput Phys Commun 87(1995)155.
- [2] R W Hockney and J W Eastwood, *Computer Simulation using Particles* (Adam-Hilger, Bristol, 1988).

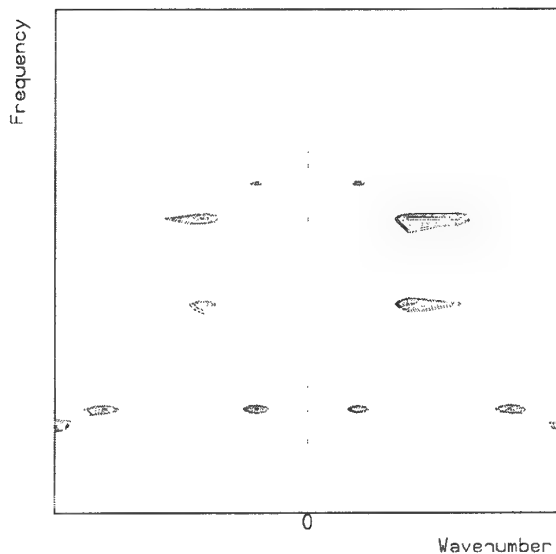


Figure 21: Dispersion test for periodic TWT geometry. The contours are of amplitude normalised structure factor $S(k, l)$ obtained from the run pc11r02. (The axis labelling has been deliberately removed to protect proprietary data.)

- [3] C K Birdsall and A B Langdon, *Plasma Physics via Computer Simulation* (Adam-Hilger, Bristol, 1991).
- [4] J W Eastwood, in Proc. 7th Ann Rev Prog App Computational Electromagnetics (Naval Postgraduate School, Monterey, Mar 1991)655.
- [5] J W Eastwood, *Computer Modelling of Microwave Sources*, in Proc 18th Annual Conf Plasma Phys (IoP, Colchester, 1991)35-42.
- [6] W Arter and J W Eastwood, *Electromagnetic Modelling in Arbitrary Geometries by the Virtual Particle Particle-Mesh Method*, in Proc 14th Int Conf Num Sim Plasmas (APS, Annapolis, Sept 1991)Paper PWE15.
- [7] W Arter, *The 3-D General Geometry PIC Software for Distributed Memory MIMD computers: Extended Description of Inputs used to Generate Meshes and Control Runs*, AEA/TYKB/31878/TN/14, October 1994.
- [8] W Arter, *The 3-D General Geometry PIC Software : The PEGGIE Pre-processor Version 2.10.00*, AEA/TYKB/28006/TN/2, September 1995.
- [9] N J Brealey, J W Eastwood and W Arter, *3DPIC Diagnostics*, AEA/TYKB/31878/TN/9, September 1994.
- [10] W Arter, *Boundary Conditions for Maxwell's Equations in General Geometry*, AEA/TYKB/31878/TN/11, September 1994.
- [11] W Arter, *Dispersion and Stability analysis for Maxwell's Equations*, AEA/TYKB/31878/TN/8, September 1994.

- [12] J W Eastwood, R W Hockney and W Arter, *General geometry PIC for MIMD computers: Final report*, Report RFFX(93)56, Culham Laboratory, June 1993.
- [13] J P Boris, *Relativistic Plasma Simulation-Optimization of a Hybrid Code*, in Proc 4th Conf Num Sim Plasmas (NRL, WA 1971)3.
- [14] W Arter, *The System of Dimensionless Units*, AEA/TYKB/31878/TN/7, September 1994.
- [15] J W Eastwood, W Arter, N J Brealey and R W Hockney, *Demonstration Calculations of HPM Sources using 3-D Electromagnetic PIC Software*, AEA/TYKB/31777/RP/1, May 1995.
- [16] N J Brealey, *MPICTIM Release 2.10.00: User Guide*, Technical Note AEA/TYKB/28006/TN/5, Culham Laboratory, September 1995.
- [17] J D Lawson, *The Physics of Charged-Particle Beams, Second Edition* (Clarendon Press, Oxford, 1988).

Appendices

A PEGGIE test data examples

05/09/13
08:11:19

bean32.uif

1

```
o_bean32
'NLRES L False for NEWRUN,True for RESET' F
bean32
beam spreading test

CHLAB3      which will appear at the start
CHLAB4      of the NOUT channel output
'CHLAB5 A **Label available to programmer          1.1 ' /
'NDIARY I **Channel for diary                      1.2 ' /
'NIN I **Current input channel                    1.2 ' /
'NLEDGE I **Channel for restart records            1.2 ' /
'NONLIN I **Channel for input-output               1.2 ' /
'NOUT I **Current output channel                   1.2 ' /
'NPRINT I **Channel for printed output             1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number                    1.2 ' /
'NRUN I **Maximum number of steps                  1.2 ' /
'NADUMP IA **Codes for array dumps                  1.9 ' /
'NPDUMP IA **Codes for dumping points               1.9 ' /
'NVDUMP IA **Codes for dumping arrays               1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector            1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector            1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector            1.9 ' /
'NLREPT L **.TRUE. if report required               1.9 ' /
'NLGRAF L **.display device geometry                7.3 ' F/

SF courant number
&SF DTMUL=1.0,
NRUN=1000,
CBLGLB='b1',XYZGLB=-0.0004949745,-0.0004949745,0.,
NLDUMP=F,
NXPTDD=5/
PS gst pcles0 alla times1
FS pcles0 pplot0
FL pplot0
&FL CFLNAM='y(x)',LFSKIP=10,LFCOLO=3,FXMIN=-0.001,FXMAX=0.001,FMIN=-0.001,FMAX=0.001/
PS gst pcles1 alla times1
FS pcles1 pplot1
FL pplot1
&FL CFLNAM='r(z)',LFSKIP=10,LFCOLO=3,FXMIN=0.0,FXMAX=0.016,FMIN=0.000,FMAX=0.001/
DS alla alldo
DO alldo b1s b4s b3s b2s b5s b1as b4as b3as b2as b5as b1bs b4bs b3bs b2bs b5bs b1cs b4cs b3cs b2cs b5cs
SU b1s b1
SU b4s b4
SU b3s b3
SU b2s b2
SU b5s b5
SU b1as b1a
SU b4as b4a
SU b3as b3a
SU b2as b2a
SU b5as b5a
SU b1bs b1b
SU b4bs b4b
SU b3bs b3b
SU b2bs b2b
SU b5bs b5b
SU b1cs b1c
SU b4cs b4c
SU b3cs b3c
SU b2cs b2c
SU b5cs b5c
TS times1 online
BG part2 polar_to_rectangular_transition -8 -8 -6
&POLRCT RADCUR=0.001,RADSTR=0.0007,THEMIN=30.,THEMAX=120.,AXMIN=0.,AXMAX=0.004/
BG cube regular_cubic_lattice -20 -20 -40
&CUBREG RZMAX=0.004,RXMAX=0.000989949,RYMAX=0.000989949/
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL b1 cube uniform1
BL b2 part2 uniform1
BL b3 sameas b2
BL b4 sameas b2
BL b5 sameas b2
BL b1a sameas b1
```

95/09/13
08:11:19

bean32.uif

2

```
BL b2a sameas b2
BL b3a sameas b2
BL b4a sameas b2
BL b5a sameas b2
BL b1b sameas b1
BL b2b sameas b2
BL b3b sameas b2
BL b4b sameas b2
BL b5b sameas b2
BL b1c sameas b1
BL b2c sameas b2
BL b3c sameas b2
BL b4c sameas b2
BL b5c sameas b2
PA pbcon pbinj pcond
SP electrons extra 1 -1 6 3 4 5
&PCLE NPBGE0=1/
PP pbinj inject_beam
&PARBCS CSPECI='electrons',VBEAM=7.085E+3,
RADBEA(2)=0.0003,ROTBEA=0.,CURBEA=-0.21/
SG inject ramp2 signalcpt2
SC ramp2
&SIGNAL
  FREQUENCY = 0.0,
  DURATION = 6.05E-12,
  POWER = 0.0 1.0,
  SIGNAL_FORM='LINEAR'/
SC signalcpt2
&SIGNAL
  FREQUENCY = 0.0,
  DURATION = 10000.,
  POWER = 1.0 1.0,
  SIGNAL_FORM='CONSTANT'/
PP pcond perfect_conductor
PA pcond sameas pcond
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=1.0 /
PA reswal sameas reswal
BC b2
W=b1:E
S=b5:N
N=b3:S
E=pcond
U=b2a:D
D=pcond
EN
BC b3
W=b1:N(1,0)(0,1)
S=b2:N
N=b4:S
E=pcond
U=b3a:D
D=pcond
EN
BC b4
W=b1:W(1,0)(0,1)
S=b3:N
N=b5:S
E=pcond
U=b4a:D
D=pcond
EN
BC b5
W=b1:S
S=b4:N
N=b2:S
E=pcond
U=b5a:D
D=pcond
EN
BC b1
S=b5:W
E=b2:W
W(1,0)(0,1)=b4:W
N(1,0)(0,1)=b3:W
D=pbcon
U=b1a:D
EN
```

25/09/13
08:11:19

bean32.uif

3

```
BC b2a
W= b1a:E
S=b5a:N
N=b3a:S
E=pcond
U=b2b:D
D=b2:U
EN
BC b3a
W=b1a:N(1,0)(0,1)
S=b2a:N
N=b4a:S
E=pcond
U=b3b:D
D=b3:U
EN
BC b4a
W=b1a:W(1,0)(0,1)
S=b3a:N
N=b5a:S
E=pcond
U=b4b:D
D=b4:U
EN
BC b5a
W=b1a:S
S=b4a:N
N=b2a:S
E=pcond
U=b5b:D
D=b5:U
EN
BC b1a
S=b5a:W
E=b2a:W
W(1,0)(0,1)=b4a:W
N(1,0)(0,1)=b3a:W
D=b1:U
U=b1b:D
EN
BC b2b
W= b1b:E
S=b5b:N
N=b3b:S
E=pcond
U=b2c:D
D=b2a:U
EN
BC b3b
W=b1b:N(1,0)(0,1)
S=b2b:N
N=b4b:S
E=pcond
U=b3c:D
D=b3a:U
EN
BC b4b
W=b1b:W(1,0)(0,1)
S=b3b:N
N=b5b:S
E=pcond
U=b4c:D
D=b4a:U
EN
BC b5b
W=b1b:S
S=b4b:N
N=b2b:S
E=pcond
U=b5c:D
D=b5a:U
EN
BC b1b
S=b5b:W
E=b2b:W
W(1,0)(0,1)=b4b:W
N(1,0)(0,1)=b3b:W
D=b1a:U
```

05/09/13
08:11:19

bean32.uif

4

```
U=b1c:D
EN
BC b2c
W= b1c:E
S=b5c:N
N=b3c:S
E=pcond
U=reswal
D=b2b:U
EN
BC b3c
W=b1c:N(1,0)(0,1)
S=b2c:N
N=b4c:S
E=pcond
U=reswal
D=b3b:U
EN
BC b4c
W=b1c:W(1,0)(0,1)
S=b3c:N
N=b5c:S
E=pcond
U=reswal
D=b4b:U
EN
BC b5c
W=b1c:S
S=b4c:N
N=b2c:S
E=pcond
U=reswal
D=b5b:U
EN
BC b1c
S=b5c:W
E=b2c:W
W(1,0)(0,1)=b4c:W
N(1,0)(0,1)=b3c:W
D=b1b:U
U=reswal
EN
OR b1 b1a b1b b1c b2 b3 b4 b5 b2a b3a b4a b5a b2b b3b b4b b5b b2c b3c b4c b5c
```

95/09/13
08:11:20

cyl07.uif

1

```
o_cyl07
'NLRES L False for NEWRUN,True for RESET' F
cyl07
CYLINDRICAL CAVITY
UNIFORM B 1 BLOCK ONLY
new G33E

'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/

SF courant number
&SF DTMUL=1.0,
NRUN=120,
NXPTDD=5,
NINIT=1,
BUNI(1)=1./
PS tsd ptflds points times
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt1 pt2 pt3 pt4 pt5
SU pt1 b1 1 2 3 0.0 0.0 0.0 0.0 0.0 0.0
SU pt2 b1 1 2 3 0.0 0.0 0.0 0.2 0.0 0.0 0.2
SU pt3 b1 1 2 3 0.0 0.0 0.6 0.0 0.0 0.6
SU pt4 b1 1 2 3 0.0 0.0 0.8 0.0 0.0 0.8
SU pt5 b1 1 2 3 0.0 0.0 1.0 0.0 0.0 1.0
TS times online
PS tsd avfld face times
FS avfld intD.ds intB.ds intj.ds
DS face onb1
SU onb1 b1 1 2 3 0.4 0. 0. 0.4 1. 1.
BG centre polar_with_regular_meshing -8 -8 -16
&POLREG RADINR=0.,RADOUT=0.001,THEMAX=360.,AXMAX=0.002/
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL b1 centre uniform1
PP pcond perfect_conductor
PP axis polar_axis
&BCS /
PA pcond sameas pcond
PA axis sameas axis
BC b1
E=pcond
D(1,0)(0,1)=pcond
U(1,0)(0,1)=pcond
S=b1:N
N=b1:S
W=axis
EN
OR b1
```

95/09/13
08:11:21

mlp903.uif

1

```
o_mlp903
'NLRES L False for NEWRUN,True for RESET' F
mlp903
MAGNETICALLY INSULATED LINE OSCILLATOR
9 cavities
periodic
ping limited to one cavity
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant number
&SF DTMUL=0.99,
NRUN=1053,
NINIT=-9,
CMODE=1.,
AMODE=0.15,-0.05,0.0,
NLDUMP=F,
NXPTDD=5/
PS lin Efield outerline timesp
FS Efield E1 E2 E3
DS outerline outer
CU outer inherit
&INHERI CBLOCK='b2a',
OPOINT=0.5,0.5,0.,
NDIR(1)=3,
NLXTEN=T,CUNIVS='all'/
TS timesp online 0. 5. 0.02 0. 0. 9
TS times online
PS tsd ptflds points times
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt1 pt2 pt3 pt4 pt5 pt6 pt7 pt8
SU pt1 ca2a 1 2 3 0.5 0.5 0.5 0.5 0.5 0.5
SU pt2 ca2b 1 2 3 0.5 0.5 0.5 0.5 0.5 0.5
SU pt3 ca2c 1 2 3 0.5 0.5 0.5 0.5 0.5 0.5
SU pt4 ca2 1 2 3 0.5 0.5 0.5 0.5 0.5 0.5
TS times online
PS tsd avfld face times
FS avfld intD.dS intB.dS intj.dS
DS face onb2
SU onb2 b2 1 2 3 0.4 0. 0. 0.4 1. 1.
BG centre polar_with_regular_meshing -8 -12 -10
&POLREG RADINR=0.,RADOUT=0.0375,THEMAX=360.,AXMAX=0.025/
BG outer polar_with_regular_meshing -8 -4 -16
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.04/
BG cavity polar_with_regular_meshing -16 -4 -16
&POLREG RADINR=0.075,RADOUT=0.150,THEMAX=120.,AXMAX=0.04/
BG inout polar_with_regular_meshing -8 -4 -20
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.05/
BG first polar_with_regular_meshing -8 -4 -4
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.01/
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL b2 outer uniform1
BL b3 sameas b2
BL b4 sameas b2
BL ca2 cavity uniform1
BL ca3 sameas ca2
BL ca4 sameas ca2
BL b2a inout uniform1
```

05/09/13
08:11:21

mlp903.uif

2

```
BL b3a sameas b2a
BL b4a sameas b2a
BL ca2a sameas ca2
BL ca3a sameas ca2
BL ca4a sameas ca2
BL b2b sameas b2a
BL b3b sameas b2a
BL b4b sameas b2a
BL ca2b sameas ca2
BL ca3b sameas ca2
BL ca4b sameas ca2
BL b2c sameas b2a
BL b3c sameas b2a
BL b4c sameas b2a
BL ca2c sameas ca2
BL ca3c sameas ca2
BL ca4c sameas ca2
BL b2d sameas b2a
BL b3d sameas b2a
BL b4d sameas b2a
BL ca2d sameas ca2
BL ca3d sameas ca2
BL ca4d sameas ca2
BL b2e sameas b2a
BL b3e sameas b2a
BL b4e sameas b2a
BL ca2e sameas ca2
BL ca3e sameas ca2
BL ca4e sameas ca2
BL b2f sameas b2a
BL b3f sameas b2a
BL b4f sameas b2a
BL ca2f sameas ca2
BL ca3f sameas ca2
BL ca4f sameas ca2
BL b2g sameas b2a
BL b3g sameas b2a
BL b4g sameas b2a
BL ca2g sameas ca2
BL ca3g sameas ca2
BL ca4g sameas ca2
BL b2h sameas b2a
BL b3h sameas b2a
BL b4h sameas b2a
BL ca2h sameas ca2
BL ca3h sameas ca2
BL ca4h sameas ca2
BL b2E first uniform1
BL b3E sameas b2E
BL b4E sameas b2E
PP pcond perfect_conductor
&BCS /
PP surfld surface_field
&BCS DAPLYA(1)=1.,DAPLYA(2)=0.,DAPLYA(3)=0.,
SURFZ=377.,NGLOB=0,DPOT=1000./
PA pcond sameas pcond
PA surfld sameas surfld
BC b2
W=pcond
N=b3:S
S=b4:N
E=ca2:W
D=b2h:U
Up =b2E:D
EN
BC b3
W=pcond
N=b4:S
S=b2:N
E=ca3:W
D=b3h:U
U=b3E:D
EN
BC b4
W=pcond
N=b2:S
S=b3:N
E=ca4:W
```

95/09/13
08:11:21

mlp903.uif

3

D=b4h:U
U=b4E:D
EN
BC ca2
W=b2:E
N=ca3:S
S=ca4:N
E=pcond
D=pcond
U=pcond
EN
BC ca3
W=b3:E
N=ca4:S
S=ca2:N
E=pcond
D=pcond
U=pcond
EN
BC ca4
W=b4:E
N=ca2:S
S=ca3:N
E=pcond
D=pcond
U=pcond
EN
BC b2a
W= pcond
N=b3a:S
S=b4a:N
E(0,0)(1,0.8)=ca2a:W
E(0,0.8)(1,1)=pcond
D=b2E:U
Up = b2b:D
EN
BC b3a
W=pcond
N=b4a:S
S=b2a:N
E(0,0)(1,0.8)=ca3a:W
E(0,0.8)(1,1)=pcond
D=b3E:U
U=b3b:D
EN
BC b4a
W=pcond
N=b2a:S
S=b3a:N
E(0,0)(1,0.8)=ca4a:W
E(0,0.8)(1,1)=pcond
D=b4E:U
U=b4b:D
EN
BC ca2a
W=b2a:E(0,0)(1,0.8)
N=ca3a:S
S=ca4a:N
E=pcond
D=pcond
U=pcond
EN
BC ca3a
W=b3a:E(0,0)(1,0.8)
N=ca4a:S
S=ca2a:N
E=pcond
D=pcond
U=pcond
EN
BC ca4a
W=b4a:E(0,0)(1,0.8)
N=ca2a:S
S=ca3a:N
E=pcond
D=pcond
U=pcond
EN

05/09/13
08:11:21

mlp903.uif

4

```
BC b2b
W= pcond
N=b3b:S
S=b4b:N
E(0,0)(1,0.8)=ca2b:W
E(0,0.8)(1,1)=pcond
D=b2a:U
Up = b2c:D
EN
BC b3b
W=pcond
N=b4b:S
S=b2b:N
E(0,0)(1,0.8)=ca3b:W
E(0,0.8)(1,1)=pcond
D=b3a:U
U=b3c:D
EN
BC b4b
W=pcond
N=b2b:S
S=b3b:N
E(0,0)(1,0.8)=ca4b:W
E(0,0.8)(1,1)=pcond
D=b4a:U
U=b4c:D
EN
BC ca2b
W=b2b:E(0,0)(1,0.8)
N=ca3b:S
S=ca4b:N
E=pcond
D=pcond
U=pcond
EN
BC ca3b
W=b3b:E(0,0)(1,0.8)
N=ca4b:S
S=ca2b:N
E=pcond
D=pcond
U=pcond
EN
BC ca4b
W=b4b:E(0,0)(1,0.8)
N=ca2b:S
S=ca3b:N
E=pcond
D=pcond
U=pcond
EN
BC b2c
W= pcond
N=b3c:S
S=b4c:N
E(0,0)(1,0.8)=ca2c:W
E(0,0.8)(1,1)=pcond
D=b2b:U
Up = b2d:D
EN
BC b3c
W=pcond
N=b4c:S
S=b2c:N
E(0,0)(1,0.8)=ca3c:W
E(0,0.8)(1,1)=pcond
D=b3b:U
U=b3d:D
EN
BC b4c
W=pcond
N=b2c:S
S=b3c:N
E(0,0)(1,0.8)=ca4c:W
E(0,0.8)(1,1)=pcond
D=b4b:U
U=b4d:D
EN
```

95/09/13
08:11:21

mlp903.uif

5

```
BC ca2c
W=b2c:E(0,0)(1,0.8)
N=ca3c:S
S=ca4c:N
E=pcond
D=pcond
U=pcond
EN
BC ca3c
W=b3c:E(0,0)(1,0.8)
N=ca4c:S
S=ca2c:N
E=pcond
D=pcond
U=pcond
EN
BC ca4c
W=b4c:E(0,0)(1,0.8)
N=ca2c:S
S=ca3c:N
E=pcond
D=pcond
U=pcond
EN
BC b2d
W= pcond
N=b3d:S
S=b4d:N
E(0,0)(1,0.8)=ca2d:W
E(0,0.8)(1,1)=pcond
D=b2c:U
Up = b2e:D
EN
BC b3d
W=pcond
N=b4d:S
S=b2d:N
E(0,0)(1,0.8)=ca3d:W
E(0,0.8)(1,1)=pcond
D=b3c:U
U=b3e:D
EN
BC b4d
W=pcond
N=b2d:S
S=b3d:N
E(0,0)(1,0.8)=ca4d:W
E(0,0.8)(1,1)=pcond
D=b4c:U
U=b4e:D
EN
BC ca2d
W=b2d:E(0,0)(1,0.8)
N=ca3d:S
S=ca4d:N
E=pcond
D=pcond
U=pcond
EN
BC ca3d
W=b3d:E(0,0)(1,0.8)
N=ca4d:S
S=ca2d:N
E=pcond
D=pcond
U=pcond
EN
BC ca4d
W=b4d:E(0,0)(1,0.8)
N=ca2d:S
S=ca3d:N
E=pcond
D=pcond
U=pcond
EN
BC b2e
W= pcond
N=b3e:S
```

95/09/13
08:11:21

mlp903.uif

6

```
S=b4e:N
E(0,0)(1,0.8)=ca2e:W
E(0,0.8)(1,1)=pcond
D=b2d:U
Up = b2f:D
EN
BC b3e
W=pcond
N=b4e:S
S=b2e:N
E(0,0)(1,0.8)=ca3e:W
E(0,0.8)(1,1)=pcond
D=b3d:U
U=b3f:D
EN
BC b4e
W=pcond
N=b2e:S
S=b3e:N
E(0,0)(1,0.8)=ca4e:W
E(0,0.8)(1,1)=pcond
D=b4d:U
U=b4f:D
EN
BC ca2e
W=b2e:E(0,0)(1,0.8)
N=ca3e:S
S=ca4e:N
E=pcond
D=pcond
U=pcond
EN
BC ca3e
W=b3e:E(0,0)(1,0.8)
N=ca4e:S
S=ca2e:N
E=pcond
D=pcond
U=pcond
EN
BC ca4e
W=b4e:E(0,0)(1,0.8)
N=ca2e:S
S=ca3e:N
E=pcond
D=pcond
U=pcond
EN
BC b2f
W= pcond
N=b3f:S
S=b4f:N
E(0,0)(1,0.8)=ca2f:W
E(0,0.8)(1,1)=pcond
D=b2e:U
Up = b2g:D
EN
BC b3f
W=pcond
N=b4f:S
S=b2f:N
E(0,0)(1,0.8)=ca3f:W
E(0,0.8)(1,1)=pcond
D=b3e:U
U=b3g:D
EN
BC b4f
W=pcond
N=b2f:S
S=b3f:N
E(0,0)(1,0.8)=ca4f:W
E(0,0.8)(1,1)=pcond
D=b4e:U
U=b4g:D
EN
BC ca2f
W=b2f:E(0,0)(1,0.8)
N=ca3f:S
```

95/09/13
08:11:21

mlp903.uif

7

```
S=ca4f:N
E=pcond
D=pcond
U=pcond
EN
BC ca3f
W=b3f:E(0,0)(1,0.8)
N=ca4f:S
S=ca2f:N
E=pcond
D=pcond
U=pcond
EN
BC ca4f
W=b4f:E(0,0)(1,0.8)
N=ca2f:S
S=ca3f:N
E=pcond
D=pcond
U=pcond
EN
BC b2g
W= pcond
N=b3g:S
S=b4g:N
E(0,0)(1,0.8)=ca2g:W
E(0,0.8)(1,1)=pcond
D=b2f:U
Up = b2h:D
EN
BC b3g
W=pcond
N=b4g:S
S=b2g:N
E(0,0)(1,0.8)=ca3g:W
E(0,0.8)(1,1)=pcond
D=b3f:U
U=b3h:D
EN
BC b4g
W=pcond
N=b2g:S
S=b3g:N
E(0,0)(1,0.8)=ca4g:W
E(0,0.8)(1,1)=pcond
D=b4f:U
U=b4h:D
EN
BC ca2g
W=b2g:E(0,0)(1,0.8)
N=ca3g:S
S=ca4g:N
E=pcond
D=pcond
U=pcond
EN
BC ca3g
W=b3g:E(0,0)(1,0.8)
N=ca4g:S
S=ca2g:N
E=pcond
D=pcond
U=pcond
EN
BC ca4g
W=b4g:E(0,0)(1,0.8)
N=ca2g:S
S=ca3g:N
E=pcond
D=pcond
U=pcond
EN
BC b2h
W= pcond
N=b3h:S
S=b4h:N
E(0,0)(1,0.8)=ca2h:W
E(0,0.8)(1,1)=pcond
```

95/09/13
08:11:21

mlp903.uif

8

```
D=b2g:U
Up = b2:D
EN
BC b3h
W=pcond
N=b4h:S
S=b2h:N
E(0,0)(1,0.8)=ca3h:W
E(0,0.8)(1,1)=pcond
D=b3g:U
U=b3:D
EN
BC b4h
W=pcond
N=b2h:S
S=b3h:N
E(0,0)(1,0.8)=ca4h:W
E(0,0.8)(1,1)=pcond
D=b4g:U
U=b4:D
EN
BC ca2h
W=b2h:E(0,0)(1,0.8)
N=ca3h:S
S=ca4h:N
E=pcond
D=pcond
U=pcond
EN
BC ca3h
W=b3h:E(0,0)(1,0.8)
N=ca4h:S
S=ca2h:N
E=pcond
D=pcond
U=pcond
EN
BC ca4h
W=b4h:E(0,0)(1,0.8)
N=ca2h:S
S=ca3h:N
E=pcond
D=pcond
U=pcond
EN
BC b2E
W= pcond
N=b3E:S
S=b4E:N
E=pcond
D=b2:U
Up = b2a:D
EN
BC b3E
W=pcond
N=b4E:S
S=b2E:N
E=pcond
D=b3:U
U=b3a:D
EN
BC b4E
W=pcond
N=b2E:S
S=b3E:N
E=pcond
D=b4:U
U=b4a:D
EN
OR b2 b3 b4 ca2 ca3 ca4 b2h b3h b4h ca2h ca3h ca4h b2g b3g b4g ca2g ca3g ca4g +
b2f b3f b4f ca2f ca3f ca4f b2e b3e b4e ca2e ca3e ca4e +
b2d b3d b4d ca2d ca3d ca4d +
b2c b3c b4c ca2c ca3c ca4c b2b b3b b4b ca2b ca3b ca4b b2a b3a b4a ca2a ca3a ca4a b2E b3E b4E
```

95/09/13
15:53:15

ml076.uif

1

o_ml076

'NLRES L False for NEWRUN,True for RESET' F

ml076

MAGNETICALLY INSULATED TRANSMISSION LINE

Particle emission

```
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/

SF courant number
&SF DTMUL=0.99,
NRUN=8021,
NLDUMP=T,
NXPTDD=5/
PS gst pcles alla times
FS pcles pplot1 pplot2
FL pplot1
&FL CFLNAM='r(z)',LFSKIP=10,LFCOLO=3,FXMIN=-0.2,FXMAX=0.05,FMIN=0.,FMAX=0.075/
FL pplot2
&FL CFLNAM='theta(z)',LFSKIP=10,LFCOLO=4,FXMIN=-0.2,FXMAX=0.05,FMIN=-180.,FMAX=180./
DS alla alldo
DO alldo b1s b2s b3s b4s
SU b1s b1
SU b2s b2
SU b3s b3
SU b4s b4
PS tsd ptflds points times2
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt1 pt2 pt3 pt4 pt5
SU pt1 b2 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt2 b2 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt3 b2 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt4 b2 1 2 3 0.5 0.5 0.8 0.5 0.5 0.8
SU pt5 b2 1 2 3 0.5 0.5 0.95 0.5 0.5 0.95
TS times online 0 10 1 0 0 9
TS times2 online
PS grl Efield outerline times
FS Efield E1 E2 E3
DS outerline outer
CU outer inherit
&INHERI CBLOCK='b2',
OPOINT=0.,0.,0.5,
NDIR(1)=2,
NLXTEN=T,CUNIVS='all'/
PS tsd avfld face times2
FS avfld intD.ds intB.ds intj.ds
DS face onb2
SU onb2 b2 1 2 3 0.4 0. 0. 0.4 1. 1.
BG centre polar_with_regular_meshing -15 -12 -6
&POLREG RADINR=0.,RADOUT=0.0375,THEMAX=360.,AXMAX=0.015/
BG outer polar_with_regular_meshing -15 -4 -80
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.2/
BP uniform1 uniform
&UNIFORM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFORM /
BL b1 centre uniform1
BL b2 outer uniform1
BL b3 sameas b2
BL b4 sameas b2
```

95/09/13
15:53:15

ml076.uif

2

```
PP pcond perfect_conductor
&BCS /
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=1.0 /
PP aplfld applied_field
&BCS DAPLYA(1)=-1.,DAPLYA(2)=0.,DAPLYA(3)=0.,
DPOT=0.5E+6,SFORM='boundary'/
PP axis polar_axis
&BCS /
SP electrons any 1 -1 5 9 8 1
PP emitter emitter
&PARBCS CSPECI='electrons',EFFE=0.0,ALFA=0.5/
PP emitter2 emitter2
&PARBCS CSPECI='electrons',EFFE=1.0,ALFA=0.5/
PA pcond sameas pcond
PA reswal sameas reswal
PA aplfld sameas aplfld
PA axis sameas axis
PA pemcon emitter pcond
PA pemcon2 emitter2 pcond
SG boundary ramp signalcpt
SC ramp
&SIGNAL
  FREQUENCY = 0.0,
  DURATION = 0.3E-9,
  POWER = 0.0 1.0,
  SIGNAL_FORM='LINEAR'/
SC signalcpt
&SIGNAL
  DURATION = 10000.,
  POWER = 1.0 1.0,
  SIGNAL_FORM='CONSTANT'/
BC b2
W(0,0)(1,0.05)=pcond
W(0,0.05)(1,0.9)=pemcon
W(0,0.9)(1,0.925)=pcond
W(0,0.925)(1,1)= b1:E (0, 0)(0.333333,1)
N=b3:S
S=b4:N
E=pcond
D=aplfld
Up =pcond
EN
BC b3
W(0,0)(1,0.05)=pcond
W(0,0.05)(1,0.9)=pemcon
W(0,0.9)(1,0.925)=pcond
W(0,0.925)(1,1)=b1:E(0.333333,0)(0.666667,1)
N=b4:S
S=b2:N
E=pcond
D=aplfld
U=pcond
EN
BC b4
W(0,0)(1,0.05)=pcond
W(0,0.05)(1,0.9)=pemcon
W(0,0.9)(1,0.925)=pcond
W(0,0.925)(1,1)=b1:E(0.666667,0)(1.,1)
N=b2:S
S=b3:N
E=pcond
D=aplfld
U=pcond
EN
BC b1
E(0,0)(0.333333,1)=b2:W(0,0.925)(1,1)
E(0.333333,0)(0.666667,1)=b3:W(0,0.925)(1,1)
E(0.666667,0)(1.,1)=b4:W(0,0.925)(1,1)
D=pcond
U=pcond
S=b1:N
N=b1:S
W=axis
EN
OR b1 b2 b3 b4
W(0,0.9)(1,0.925)=pcond
```

05/09/14
10:39:04

ml407.uif

1

```
o_ml407
'NLRES L False for NEWRUN,True for RESET' F
ml407
MAGNETICALLY INSULATED LINE OSCILLATOR
4 cavities
particle emission
DPOT of 500 kv
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant number
&SF DTMUL=0.99,
NRUN=20000,
NTPRES=1,
NXPTDD=5/
PS gst pcles alla times
FS pcles pplot1 pplot2
FL pplot1
&FL CFLNAM='r(z)',LFSKIP=10,LFCOLO=3,FXMIN=-0.2,FXMAX=0.05,FMIN=0.,FMAX=0.150/
FL pplot2
&FL CFLNAM='theta(z)',LFSKIP=10,LFCOLO=4,FXMIN=-0.2,FXMAX=0.05,FMIN=-180.,FMAX=180./
DS alla alldo
DO alldo b1s b2s b3s b4s ca2s ca3s ca4s b2cs b3cs b4cs ca2cs ca3cs ca4cs +
b2bs b3bs b4bs ca2bs ca3bs ca4bs b2as b3as b4as ca2as ca3as ca4as b2Es b3Es b4Es
SU b1s b1
SU b2s b2
SU b3s b3
SU b4s b4
SU ca2s ca2
SU ca3s ca3
SU ca4s ca4
SU b2cs b2c
SU b3cs b3c
SU b4cs b4c
SU ca2cs ca2c
SU ca3cs ca3c
SU ca4cs ca4c
SU b2bs b2b
SU b3bs b3b
SU b4bs b4b
SU ca2bs ca2b
SU ca3bs ca3b
SU ca4bs ca4b
SU b2as b2a
SU b3as b3a
SU b4as b4a
SU ca2as ca2a
SU ca3as ca3a
SU ca4as ca4a
SU b2Es b2E
SU b3Es b3E
SU b4Es b4E
PS tsd ptflds points times2
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt1 pt2 pt3 pt4 pt5
SU pt1 b2E 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt2 b2a 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt3 b2c 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt4 b2 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt5 b2 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
PS tsd ptflds cavs times2
```


05/09/14
0:39:04

ml407.uif

2

```
DS cavs cav2 cav3 cav4 cav5
SU cav2 ca2a 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav3 ca2b 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav4 ca2c 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav5 ca2 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
TS times online
TS times2 online 0. 50. 0.006 0. 0. 9
PS gr1 Efield outline times
FS Efield E1 E2 E3
DS outline outer
CU outer inherit
&INHERI CBLOCK='b2b',
OPOINT=0.,0.,0.8,
NDIR(1)=2,
NLXTEN=T,CUNIVS='all'/
PS tsd avfld face times2
FS avfld intD.ds intB.ds intj.ds
DS face onb2
SU onb2 b2 1 2 3 0.4 0. 0. 0.4 1. 1.
BG centre polar_with_regular_meshing -8 -12 -10
&POLREG RADINR=0.,RADOUT=0.0375,THEMAX=360.,AXMAX=0.025/
BG outer polar_with_regular_meshing -8 -4 -16
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.04/
BG cavity polar_with_regular_meshing -16 -4 -16
&POLREG RADINR=0.075,RADOUT=0.150,THEMAX=120.,AXMAX=0.04/
BG inout polar_with_regular_meshing -8 -4 -20
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.05/
BG first polar_with_regular_meshing -8 -4 -4
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.01/
BP uniform1 uniform
&UNIFORM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFORM /
BL b1 centre uniform1
BL b2 outer uniform1
BL b3 sameas b2
BL b4 sameas b2
BL ca2 cavity uniform1
BL ca3 sameas ca2
BL ca4 sameas ca2
BL b2a inout uniform1
BL b3a sameas b2a
BL b4a sameas b2a
BL ca2a sameas ca2
BL ca3a sameas ca2
BL ca4a sameas ca2
BL b2b sameas b2a
BL b3b sameas b2a
BL b4b sameas b2a
BL ca2b sameas ca2
BL ca3b sameas ca2
BL ca4b sameas ca2
BL b2c sameas b2a
BL b3c sameas b2a
BL b4c sameas b2a
BL ca2c sameas ca2
BL ca3c sameas ca2
BL ca4c sameas ca2
BL b2E first uniform1
BL b3E sameas b2E
BL b4E sameas b2E
PP pcond perfect_conductor
&BCS /
PP reswal resistive_wall
&BCS SURFZ=377.,THETA=1.0 /
PP aplfld applied_field
&BCS DAPLYA(1)=-1.,DAPLYA(2)=0.,DAPLYA(3)=0.,
DPOT=0.5E+6,SFORM='boundary'/
PP axis polar_axis
&BCS /
PA pcond sameas pcond
PA reswal sameas reswal
PA aplfld sameas aplfld
PA axis sameas axis
PA pemcon emitter pcond
SP electrons any 1 -1 6 9 8 1
PP emitter emitter
&PARBCS CSPECI='electrons',EFFE=1.0/
```

95/09/14
10:39:04

ml407.uif

3

```
SG boundary ramp signalcpt
SC ramp
&SIGNAL
    FREQUENCY = 0.0,
    DURATION = 0.3E-9,
    POWER = 0.0, 1.0,
    SIGNAL_FORM='LINEAR'/
SC signalcpt
&SIGNAL
    DURATION = 10000.,
    POWER = 1.0, 1.0,
    SIGNAL_FORM='CONSTANT'/
BC b2
W(0,0)(1,0.375)=pemcon
W(0,0.375)(1,1)= b1:E (0, 0) (0.333333,1)
N=b3:S
S=b4:N
E=ca2:W
D=b2c:U
Up =pcond
EN
BC b3
W(0,0)(1,0.375)=pemcon
W(0,0.375)(1,1)=b1:E(0.333333,0) (0.666667,1)
N=b4:S
S=b2:N
E=ca3:W
D=b3c:U
U=pcond
EN
BC b4
W(0,0)(1,0.375)=pemcon
W(0,0.375)(1,1)=b1:E(0.666667,0) (1.,1)
N=b2:S
S=b3:N
E=ca4:W
D=b4c:U
U=pcond
EN
BC b1
E(0,0)(0.333333,1)=b2:W(0,0.375)(1,1)
E(0.333333,0)(0.666667,1)=b3:W(0,0.375)(1,1)
E(0.666667,0)(1.,1)=b4:W(0,0.375)(1,1)
D=pemcon
U=pcond
S=b1:N
N=b1:S
W=axis
EN
BC ca2
W=b2:E
N=ca3:S
S=ca4:N
E=pcond
D=pcond
U=pcond
EN
BC ca3
W=b3:E
N=ca4:S
S=ca2:N
E=pcond
D=pcond
U=pcond
EN
BC ca4
W=b4:E
N=ca2:S
S=ca3:N
E=pcond
D=pcond
U=pcond
EN
BC b2a
W= pemcon
N=b3a:S
S=b4a:N
E(0,0)(1,0.8)=ca2a:W
```

95/09/14
10:39:04

ml407.uif

4

```
E(0,0.8)(1,1)=pcond
D=b2E:U
Up = b2b:D
EN
BC b3a
W=pemcon
N=b4a:S
S=b2a:N
E(0,0)(1,0.8)=ca3a:W
E(0,0.8)(1,1)=pcond
D=b3E:U
U=b3b:D
EN
BC b4a
W=pemcon
N=b2a:S
S=b3a:N
E(0,0)(1,0.8)=ca4a:W
E(0,0.8)(1,1)=pcond
D=b4E:U
U=b4b:D
EN
BC ca2a
W=b2a:E(0,0)(1,0.8)
N=ca3a:S
S=ca4a:N
E=pcond
D=pcond
U=pcond
EN
BC ca3a
W=b3a:E(0,0)(1,0.8)
N=ca4a:S
S=ca2a:N
E=pcond
D=pcond
U=pcond
EN
BC ca4a
W=b4a:E(0,0)(1,0.8)
N=ca2a:S
S=ca3a:N
E=pcond
D=pcond
U=pcond
EN
BC b2b
W=pemcon
N=b3b:S
S=b4b:N
E(0,0)(1,0.8)=ca2b:W
E(0,0.8)(1,1)=pcond
D=b2a:U
Up = b2c:D
EN
BC b3b
W=pemcon
N=b4b:S
S=b2b:N
E(0,0)(1,0.8)=ca3b:W
E(0,0.8)(1,1)=pcond
D=b3a:U
U=b3c:D
EN
BC b4b
W=pemcon
N=b2b:S
S=b3b:N
E(0,0)(1,0.8)=ca4b:W
E(0,0.8)(1,1)=pcond
D=b4a:U
U=b4c:D
EN
BC ca2b
W=b2b:E(0,0)(1,0.8)
N=ca3b:S
S=ca4b:N
E=pcond
```

95/09/14
10:39:04

ml407.uif

5

```
D=pcond
U=pcond
EN
BC ca3b
W=b3b:E(0,0)(1,0.8)
N=ca4b:S
S=ca2b:N
E=pcond
D=pcond
U=pcond
EN
BC ca4b
W=b4b:E(0,0)(1,0.8)
N=ca2b:S
S=ca3b:N
E=pcond
D=pcond
U=pcond
EN
BC b2c
W= pemcon
N=b3c:S
S=b4c:N
E(0,0)(1,0.8)=ca2c:W
E(0,0.8)(1,1)=pcond
D=b2b:U
Up = b2:D
EN
BC b3c
W=pemcon
N=b4c:S
S=b2c:N
E(0,0)(1,0.8)=ca3c:W
E(0,0.8)(1,1)=pcond
D=b3b:U
U=b3:D
EN
BC b4c
W=pemcon
N=b2c:S
S=b3c:N
E(0,0)(1,0.8)=ca4c:W
E(0,0.8)(1,1)=pcond
D=b4b:U
U=b4:D
EN
BC ca2c
W=b2c:E(0,0)(1,0.8)
N=ca3c:S
S=ca4c:N
E=pcond
D=pcond
U=pcond
EN
BC ca3c
W=b3c:E(0,0)(1,0.8)
N=ca4c:S
S=ca2c:N
E=pcond
D=pcond
U=pcond
EN
BC ca4c
W=b4c:E(0,0)(1,0.8)
N=ca2c:S
S=ca3c:N
E=pcond
D=pcond
U=pcond
EN
BC b2E
W= pcond
N=b3E:S
S=b4E:N
E=pcond
D=aplfld
Up = b2a:D
EN
```

05/09/14
10:39:04

ml407.uif

6

BC b3E
W=pcond
N=b4E:S
S=b2E:N
E=pcond
D=aplfld
U=b3a:D
EN
BC b4E
W=pcond
N=b2E:S
S=b3E:N
E=pcond
D=aplfld
U=b4a:D
EN
OR b1 b2 b3 b4 ca2 ca3 ca4 b2c b3c b4c ca2c ca3c ca4c b2b b3b b4b ca2b ca3b ca4b b2a b3a b4a ca2a ca3a ca4a b2E

95/09/13
08:11:26

porb21.uif

1

```
o_porb21
'NLRES L False for NEWRUN,True for RESET' F
porb21
Uniform B
All orientations of cubes
particle orbit test
z cpt of velocity
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant_number
&SF DTMUL=1.0,
NRUN=400,
NLDUMP=F,
MALG(19)=1,MALG(20)=1,
NXPTDD=5/
PS gst pcles0 alla times1
FS pcles0 pplot0
FL pplot0
&FL CFLNAM='y(x)',LFSKIP=1,LFCOLO=3,FXMIN=-0.005,FXMAX=0.025,FMIN=-0.01,FMAX=0.02/
PS gst pcles1 alla times1
FS pcles1 pplot1
FL pplot1
&FL CFLNAM='x(z)',LFSKIP=1,LFCOLO=3,FXMIN=0.0,FXMAX=0.02,FMIN=0.0,FMAX=0.02/
DS alla alldo
DO alldo b1s b2s b3s b4s b5s b6s b7s b8s
SU b1s b1
SU b2s b2
SU b3s b3
SU b4s b4
SU b5s b5
SU b6s b6
SU b7s b7
SU b8s b8
PS tsd ptflds points times2
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt2 pt3 pt4 pt5 cav3 cav4 cav5
SU pt2 b8 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt3 b4 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt4 b2 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt5 b2 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU cav3 b7 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav4 b5 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav5 b3 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
TS times1 online 0 1000 1 0 0 0
TS times2 online 0 1000 1 0 0 0
BG type1 regular_cubic_lattice -4 -4 -4
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BG type2 regular_cubic_lattice -4 -4 -4
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL b1 type1 uniform1
BL b2 type2 uniform1
BL b3 sameas b1
BL b4 sameas b1
BL b5 sameas b1
BL b6 sameas b1
BL b7 sameas b1
```

15/09/13
08:11:26

porb21.uif

2

```
BL b8 sameas b1
SP electrons below 1 -1 0 0 8 1
&PCLE CBLOCK='b1',
UCPOS=0.218,0.1,0.5, -1.,
VEL=0.,-1.E+8,1.5E+7,NCVEL=1/
PO bext
&SF NINIT=1,BUNI=0.0,0.0,-0.054/
PP pcond perfect_conductor
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=0.5 /
PP aplfld applied_field
&BCS DAPLYA(1)=115.5712,DAPLYA(2)=0.,DAPLYA(3)=0. /
PA pcond sameas pcond
PA reswal sameas reswal
PA aplfld sameas aplfld
BC b1
D=b5:N(1,0)(0,1)
S=b4:D
W=b2:N(1,0)(0,1)
E=b2:S(1,0)(0,1)
N=b4:U
U=b5:S(1,0)(0,1)
EN
BC b2
D=b6:U(0,1)(1,0)
S(1,0)(0,1)=b1:E
W=b3:S(0,1)(1,0)
E=b3:N(0,1)(1,0)
N(1,0)(0,1)=b1:W
U=b6:D(0,1)(1,0)
EN
BC b3
D=b4:W(0,1)(1,0)
S(0,1)(1,0)=b2:W
W=b7:E
E=b7:W
N(0,1)(1,0)=b2:E
U=b4:E(0,1)(1,0)
EN
BC b4
D=b1:S
S=b8:S(1,0)(0,1)
W(0,1)(1,0)=b3:D
E(0,1)(1,0)=b3:U
N=b8:N(1,0)(0,1)
U=b1:N
EN
BC b5
D=b8:D(0,1)(1,0)
S(1,0)(0,1)=b1:U
W=b6:W
E=b6:E
N(1,0)(0,1)=b1:D
U=b8:U(0,1)(1,0)
EN
BC b6
D(0,1)(1,0)=b2:U
S=b7:N(0,1)(1,0)
W=b5:W
E=b5:E
N=b7:S(0,1)(1,0)
U(0,1)(1,0)=b2:D
EN
BC b7
D=b8:E(1,1)(0,0)
S(0,1)(1,0)=b6:N
W=b3:E
E=b3:W
N(0,1)(1,0)=b6:S
U=b8:W(1,1)(0,0)
EN
BC b8
D(0,1)(1,0)=b5:D
S(1,0)(0,1)=b4:S
W(1,1)(0,0)=b7:U
E(1,1)(0,0)=b7:D
N(1,0)(0,1)=b4:N
U(0,1)(1,0)=b5:U
```

95/09/13
08:11:26

porb21.uif

3

EN
OR b1 b2 b3 b4 b5 b6 b7 b8

05/09/13
15:53:13

porb35.uif

1

```
o_porb35
'NLRES L False for NEWRUN,True for RESET' F
porb35
Uniform B
All orientations of cubes
particle orbit test
B in y direction
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant_number
&SF DTMUL=1.0,
NRUN=400,
NLDUMP=F,
NINPOR(19)=1,NINPOR(20)=1,
NXPTDD=5/
PS gst pcles0 alla times1
FS pcles0 pplot0
FL pplot0
&FL CFLNAM='z(x)',LFSKIP=1,LFCOLO=3,FXMIN=0.0,FXMAX=0.02,FMIN=0.0,FMAX=0.02/
PS gst pcles1 alla times1
FS pcles1 pplot1
FL pplot1
&FL CFLNAM='y(x)',LFSKIP=1,LFCOLO=3,FXMIN=-0.005,FXMAX=0.025,FMIN=-0.01,FMAX=0.02/
DS alla alldo
DO alldo b1s b2s b3s b4s b5s b6s b7s b8s
SU b1s b1
SU b2s b2
SU b3s b3
SU b4s b4
SU b5s b5
SU b6s b6
SU b7s b7
SU b8s b8
PS tsd ptflds points times2
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt2 pt3 pt4 pt5 cav3 cav4 cav5
SU pt2 b8 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt3 b4 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt4 b2 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt5 b2 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU cav3 b7 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav4 b5 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav5 b3 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
TS times1 online 0 1000 1 0 0 0
TS times2 online 0 1000 1 0 0 0
BG type1 regular_cubic_lattice -4 -4 -4
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BG type2 regular_cubic_lattice -4 -4 -4
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BP uniform1 uniform
&UNIFORM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFORM /
BL b1 type1 uniform1
BL b2 type2 uniform1
BL b3 sameas b1
BL b4 sameas b1
BL b5 sameas b1
BL b6 sameas b1
BL b7 sameas b1
```

05/09/13
15:53:13

porb35.uif

2

```
BL b8 sameas b1
SP electrons below 1 -1 0 0 8 1
&PCLE CBLOCK='b1',
UCPOS=0.218,0.5,0.95, -1.,
VEL=0.,1.5E+7,-1.E+8,0.,NCVEL=1/
PO bext
&SF NINIT=1,BUNI=0.0,0.054,0.0/
PP pcond perfect_conductor
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=0.5 /
PP aplfld applied_field
&BCS DAPLYA(1)=115.5712,DAPLYA(2)=0.,DAPLYA(3)=0. /
PA pcond sameas pcond
PA reswal sameas reswal
PA aplfld sameas aplfld
BC b1
D=b5:N(1,0)(0,1)
S=b4:D
W=b2:N(1,0)(0,1)
E=b2:S(1,0)(0,1)
N=b4:U
U=b5:S(1,0)(0,1)
EN
BC b2
D=b6:U(0,1)(1,0)
S(1,0)(0,1)=b1:E
W=b3:S(0,1)(1,0)
E=b3:N(0,1)(1,0)
N(1,0)(0,1)=b1:W
U=b6:D(0,1)(1,0)
EN
BC b3
D=b4:W(0,1)(1,0)
S(0,1)(1,0)=b2:W
W=b7:E
E=b7:W
N(0,1)(1,0)=b2:E
U=b4:E(0,1)(1,0)
EN
BC b4
D=b1:S
S=b8:S(1,0)(0,1)
W(0,1)(1,0)=b3:D
E(0,1)(1,0)=b3:U
N=b8:N(1,0)(0,1)
U=b1:N
EN
BC b5
D=b8:D(0,1)(1,0)
S(1,0)(0,1)=b1:U
W=b6:W
E=b6:E
N(1,0)(0,1)=b1:D
U=b8:U(0,1)(1,0)
EN
BC b6
D(0,1)(1,0)=b2:U
S=b7:N(0,1)(1,0)
W=b5:W
E=b5:E
N=b7:S(0,1)(1,0)
U(0,1)(1,0)=b2:D
EN
BC b7
D=b8:E(1,1)(0,0)
S(0,1)(1,0)=b6:N
W=b3:E
E=b3:W
N(0,1)(1,0)=b6:S
U=b8:W(1,1)(0,0)
EN
BC b8
D(0,1)(1,0)=b5:D
S(1,0)(0,1)=b4:S
W(1,1)(0,0)=b7:U
E(1,1)(0,0)=b7:D
N(1,0)(0,1)=b4:N
U(0,1)(1,0)=b5:U
```

05/09/13
15:53:13

porb35.uif

3

EN

OR b1 b2 b3 b4 b5 b6 b7 b8

05/09/13
15:53:14

porb40.uif

1

```
o_porb40
'NLRES L False for NEWRUN,True for RESET' F
porb40
Uniform B
All orientations of cubes
particle orbit test
field in x direction
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant_number
&SF DTMUL=1.0,
NRUN=400,
NLDUMP=F,
NINPOR(19)=1,NINPOR(20)=1,
NXPTDD=5/
PS gst pcles0 alla times1
FS pcles0 pplot0
FL pplot0
&FL CFLNAM='y(z)',LFSKIP=1,LFCOLO=3,FXMIN=-0.005,FXMAX=0.025,FMIN=-0.01,FMAX=0.02/
PS gst pcles1 alla times1
FS pcles1 pplot1
FL pplot1
&FL CFLNAM='x(z)',LFSKIP=1,LFCOLO=3,FXMIN=0.0,FXMAX=0.02,FMIN=0.0,FMAX=0.02/
DS alla alldo
DO alldo b1s b2s b3s b4s b5s b6s b7s b8s
SU b1s b1
SU b2s b2
SU b3s b3
SU b4s b4
SU b5s b5
SU b6s b6
SU b7s b7
SU b8s b8
PS tsd ptflds points times2
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt2 pt3 pt4 pt5 cav3 cav4 cav5
SU pt2 b8 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt3 b4 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt4 b2 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt5 b2 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU cav3 b7 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav4 b5 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav5 b3 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
TS times1 online 0 1000 1 0 0 0
TS times2 online 0 1000 1 0 0 0
BG type1 regular_cubic_lattice -4 -4 -4
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BG type2 regular_cubic_lattice -4 -4 -4
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BP uniform1 uniform
&UNIFORM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFORM /
BL b1 type1 uniform1
BL b2 type2 uniform1
BL b3 sameas b1
BL b4 sameas b1
BL b5 sameas b1
BL b6 sameas b1
BL b7 sameas b1
```

05/09/13
15:53:14

porb40.uif

2

```
BL b8 sameas b1
SP electrons below 1 -1 0 0 8 1
&PCLE CBLOCK='b1',
UCPOS=0.5,1.,0.118, -1.,
VEL=1.5E+7,-1.E+8,0.,NCVEL=1/
PO bext
&SF NINIT=1,BUNI=0.054,0.0,0.0/
PP pcond perfect_conductor
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=0.5 /
PP aplfld applied_field
&BCS DAPLYA(1)=115.5712,DAPLYA(2)=0.,DAPLYA(3)=0. /
PA pcond sameas pcond
PA reswal sameas reswal
PA aplfld sameas aplfld
BC b1
D=b5:N(1,0)(0,1)
S=b4:D
W=b2:N(1,0)(0,1)
E=b2:S(1,0)(0,1)
N=b4:U
U=b5:S(1,0)(0,1)
EN
BC b2
D=b6:U(0,1)(1,0)
S(1,0)(0,1)=b1:E
W=b3:S(0,1)(1,0)
E=b3:N(0,1)(1,0)
N(1,0)(0,1)=b1:W
U=b6:D(0,1)(1,0)
EN
BC b3
D=b4:W(0,1)(1,0)
S(0,1)(1,0)=b2:W
W=b7:E
E=b7:W
N(0,1)(1,0)=b2:E
U=b4:E(0,1)(1,0)
EN
BC b4
D=b1:S
S=b8:S(1,0)(0,1)
W(0,1)(1,0)=b3:D
E(0,1)(1,0)=b3:U
N=b8:N(1,0)(0,1)
U=b1:N
EN
BC b5
D=b8:D(0,1)(1,0)
S(1,0)(0,1)=b1:U
W=b6:W
E=b6:E
N(1,0)(0,1)=b1:D
U=b8:U(0,1)(1,0)
EN
BC b6
D(0,1)(1,0)=b2:U
S=b7:N(0,1)(1,0)
W=b5:W
E=b5:E
N=b7:S(0,1)(1,0)
U(0,1)(1,0)=b2:D
EN
BC b7
D=b8:E(1,1)(0,0)
S(0,1)(1,0)=b6:N
W=b3:E
E=b3:W
N(0,1)(1,0)=b6:S
U=b8:W(1,1)(0,0)
EN
BC b8
D(0,1)(1,0)=b5:D
S(1,0)(0,1)=b4:S
W(1,1)(0,0)=b7:U
E(1,1)(0,0)=b7:D
N(1,0)(0,1)=b4:N
U(0,1)(1,0)=b5:U
```

95/09/13
15:53:14

porb40.uif

3

EN
OR b1 b2 b3 b4 b5 b6 b7 b8

95/09/13
08:11:27

porg60h.uif



```
o_porg60h
'NLRES L False for NEWRUN,True for RESET' F
porg60h
weights one half
subcycling 10x
CHLAB3 which will appear at the start
CHLAB4 of the NOUT channel output
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant number
&SF DTMUL=1.0,
NRUN=400,
CBLGLB='bla',XYZGLB=0.,0.0005,0.,
MALG(19)=1,MALG(20)=1,
NLDUMP=F,
NXPTDD=5/
PS pts pcles0 alla times1
FS pcles0 pplot0
FL pplot0
&FL CFLNAM='y(x)',LFSKIP=1,LFCOLO=3,FXMIN=-0.001,FXMAX=0.001,FMIN=-0.001,FMAX=0.001/
PS pts pcles1 alla times1
FS pcles1 pplot1
FL pplot1
&FL CFLNAM='x(z)',LFSKIP=1,LFCOLO=3,FXMIN=0.0,FXMAX=0.002,FMIN=-0.001,FMAX=0.001/
DS alla alldo
DO alldo blas b1bs b1cs b4s b3s b2s b7s b6s b5s
SU blas bla
SU b1bs b1b
SU b1cs b1c
SU b4s b4
SU b3s b3
SU b2s b2
SU b7s b7
SU b6s b6
SU b5s b5
TS times1 online 0 1000 1 0 0 0
BG part2 polar_to_rectangular_transition -4 -4 -6
&POLRCT RADCUR=0.001,RADSTR=0.0005,THEMIN=30.,THEMAX=90.,AXMIN=0.,AXMAX=0.002/
BG part1 parallelepiped -4 -4 -6
&PIPED RSID3=0.002,RSID1=0.0005,RSID2=0.0005,THET1=90.,THET2=90.,
RPHI1=-90.,RPHI2=-30./
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL bla part1 uniform1
BL b1b sameas bla
BL b1c sameas bla
BL b2 part2 uniform1
BL b3 sameas b2
BL b4 sameas b2
BL b5 sameas b2
BL b6 sameas b2
BL b7 sameas b2
SP electrons below 1 -1 0 0 8 10
&PCLE NPBGE0=1,CBLOCK='b1b',
UCPOS=1.,0.559,0.5, -1.,
VEL=1.E+8,0.,0.,NCVEL=1/
SP eb below 1 -1 0 0 8 10
&PCLE CBLOCK='b5',
```

95/09/13
08:11:27

porg60h.uif

2

```
MALG(3)=10,
UCPOS=0.5,1.0,0.5, -1.,
VEL=1.E+8,0.,0.,NCVEL=1/
PO bfield
&SF NINIT=1,BUNI=0.0,0.0,2.16/
PP pcond perfect_conductor
PA pcond sameas pcond
BC b2
W(1,0)(0,1)= b1a:W
S=b3:N
N=b7:S
E=pcond
U=b2:D
D=b2:U
EN
BC b3
W(1,0)(0,1)=b1a:N
S=b4:N
N=b2:S
E=pcond
U=b3:D
D=b3:U
EN
BC b4
W(1,0)(0,1)=b1c:W
S=b5:N
N=b3:S
E=pcond
U=b4:D
D=b4:U
EN
BC b5
W(1,0)(0,1)=b1c:N
S=b6:N
N=b4:S
E=pcond
U=b5:D
D=b5:U
EN
BC b6
W(1,0)(0,1)=b1b:W
S=b7:N
N=b5:S
E=pcond
U=b6:D
D=b6:U
EN
BC b7
W(1,0)(0,1)=b1b:N
S=b2:N
N=b6:S
E=pcond
U=b7:D
D=b7:U
EN
BC b1a
S=b1b:E(1,0)(0,1)
E=b1c:S(1,0)(0,1)
W=b2:W(1,0)(0,1)
N=b3:W(1,0)(0,1)
D(1,0)(0,1)=b1a:U(1,0)(0,1)
U(1,0)(0,1)=b1a:D(1,0)(0,1)
EN
BC b1c
S(1,0)(0,1)=b1a:E
E=b1b:S(1,0)(0,1)
W=b4:W(1,0)(0,1)
N=b5:W(1,0)(0,1)
D(1,0)(0,1)=b1c:U(1,0)(0,1)
U(1,0)(0,1)=b1c:D(1,0)(0,1)
EN
BC b1b
S(1,0)(0,1)=b1c:E
E(1,0)(0,1)=b1a:S
W=b6:W(1,0)(0,1)
N=b7:W(1,0)(0,1)
D(1,0)(0,1)=b1b:U(1,0)(0,1)
U(1,0)(0,1)=b1b:D(1,0)(0,1)
```


95/09/13
08:11:27

porg60h.uif

3

EN

OR b1a b1b b1c b4 b3 b2 b7 b6 b5

95/09/13
08:11:28

porg70.uif

1

```
o_porg70
'NLRES L False for NEWRUN,True for RESET' F
porg70
general geometry
periodic case
CHLAB3 which will appear at the start
CHLAB4 of the NOUT channel output
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant_number
&SF DTMUL=1.0,
NRUN=400,
NXPTDD=5,
MALG(19)=1,MALG(20)=1,
NLDUMP=F,
NXPTDD=5/
PS pts pcles0 alla times1
FS pcles0 pplot0
FL pplot0
&FL CFLNAM='y(x)',LFSKIP=1,LFCOLO=3,FXMIN=0.,FXMAX=0.002,FMIN=0.,FMAX=0.002/
PS pts pcles1 alla times1
FS pcles1 pplot1
FL pplot1
&FL CFLNAM='x(z)',LFSKIP=1,LFCOLO=3,FXMIN=0.0,FXMAX=0.005,FMIN=0.,FMAX=0.002/
DS alla alldo
DO alldo bls
SU bls bl
TS times1 online 0 1000 1 0 0 0
BG type1 parallelepiped -10 -10 -10
&PIPED RSID1=2.236068E-3,RSID2=2.4494897E-3,RSID3=1.E-3,
THET1=26.56505,THET2=35.26439,RPHI1=0.,RPHI2=45/
BP uniform1 uniform
&UNIFORM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFORM /
BL bl type1 uniform1
SP electrons below 1 -1 0 0 8 10
&PCLE NPBGEO=1,CBLOCK='b1',
UCPOS=0.5,0.5,0.5, -1.,
VEL=1.E+8,0.,0.,NCVEL=1/
SP eb below 1 -1 0 0 8 10
&PCLE CBLOCK='b1',
UCPOS=0.5,0.5,0.5, -1.,
VEL=1.E+8,0.,1.5E+7,NCVEL=1/
PO bfield
&SF NINIT=1,BUNI=0.0,0.0,2.16/
BC bl
E=b1:W
D=b1:U
U=b1:D
S=b1:N
N=b1:S
W=b1:E
EN
OR bl
```

95/09/13
08:11:29

sbc16z.uif



```
o_sbc16z
'NLRES L False for NEWRUN,True for RESET' F
sbc16z
TM mode in cube as bc

'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/

SF courant_number
&SF DTMUL=1.0,
NINIT=5,
NRUN=200,
NXPTDD=5,
AMODE=0.01,0.01,0.01
CMODE=1.
NMODE=1,1,1/
PS tsd ptflds points times
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt1 pt2 pt3 pt4 pt5 pt6 pt1t pt2t pt3t pt4t pt5t pt6t
SU pt1 b1 1 2 3 0.5 0.5 0. 0.5 0.5 0.
SU pt2 b2 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt3 b3 1 2 3 0.5 0.5 0.4 0.5 0.5 0.4
SU pt4 b4 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt5 b5 1 2 3 0.5 0.5 0.8 0.5 0.5 0.8
SU pt6 b6 1 2 3 0.5 0.5 1.0 0.5 0.5 1.0
SU pt1t b1 1 2 3 0.0 0.5 0. 0.5 0.5 0.
SU pt2t b2 1 2 3 0.0 0.5 0.2 0.5 0.5 0.2
SU pt3t b3 1 2 3 0.0 0.5 0.4 0.5 0.5 0.4
SU pt4t b4 1 2 3 0.0 0.5 0.6 0.5 0.5 0.6
SU pt5t b5 1 2 3 0.0 0.5 0.8 0.5 0.5 0.8
SU pt6t b6 1 2 3 0.0 0.5 1.0 0.5 0.5 1.0
TS times online
PS tsd avfld face times2
FS avfld intD.ds intB.ds intj.ds
DS face onb1 onb2 onb3 onb4
SU onb1 b1 1 2 3 1. 0. 0. 1. 1. 1.
TS times2 online 0. 10. 0.002 0. 0. 9
BG type1 regular_cubic_lattice -6 -8 -10
&CUBREG RXMAX=0.01,RYMAX=0.01,RZMAX=0.01/
BP uniform1 uniform
&UNIFORM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFORM /
BL b1 type1 uniform1
PP pcond perfect_conductor
PP reswal resistive_wall
&BCS SURFZ=217.51,STHETA=0.5 /
SG boundary signalcpt
SC signalcpt
&SIGNAL FREQUENCY=2.598E+10,DURATION=100.,SIGNAL_FORM='CONSTANT'/
PP surfld surface_field
&BCS DAPLYA(1)=0.0,DAPLYA(2)=0.0,DAPLYA(3)=1.0,
EAMP=1000.,NGLOB=1,SFORM='boundary'/
PA pcond sameas pcond
PA reswal sameas reswal
PA surfld sameas surfld
BC b1
E=pcond
D=surfld
```

95/09/13
08:11:29

U=reswal
S=pcond
N=pcond
W=pcond
EN
OR b1

sbc16z.uif

2

95/09/13
08:11:31

tp06.uif

1

```
o_tp06
'NLRES L False for NEWRUN,True for RESET' F
tp06
TE111 mode
periodic case
cylinder with parallelogram cross-section

'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/

SF courant_number
&SF DTMUL=1.0,
NINIT=4,
NRUN=200,
NXPTDD=5,
AMODE=1.,1.,1.
CMODE=1.
NMODE=1,1,1/
PS tsd ptflds points times
FS ptflds E1 E2 E3 H1 H2 H3
DS points bitb1
SU bitb1 b1 1 2 3 0.5 0.5 0.5 0.5 0.5 0.5
TS times online
PS tsd avfld face times2
FS avfld intD.ds intB.ds intj.ds
DS face onb1
SU onb1 b1 1 2 3 1. 0. 0. 1. 1. 1.
TS times2 online 0. 10. 0.2 0. 0. 9
BG type1 quadrilateral_cylinder -10 -10 -10
&QUADRI XQUAD1=2.,YQUAD1=2.,XQUAD2=4.,YQUAD2=2.,
RXMAX=2.,RYMAX=0.,RZMAX=2./
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL b1 type1 uniform1
BC b1
E=b1:W
D=b1:U
U=b1:D
S=b1:N
N=b1:S
W=b1:E
EN
OR b1
```

95/09/13
08:11:33

une51.uif

1

```
o_une51
'NLRES L False for NEWRUN,True for RESET' F
une51
TE111 mode
All orientations of cubes
CHLAB3 which will appear at the start
CHLAB4 of the NOUT channel output
'CHLAB5 A **Label available to programmer 1.1 ' /
'NDIARY I **Channel for diary 1.2 ' /
'NIN I **Current input channel 1.2 ' /
'NLEDGE I **Channel for restart records 1.2 ' /
'NONLIN I **Channel for input-output 1.2 ' /
'NOUT I **Current output channel 1.2 ' /
'NPRINT I **Channel for printed output 1.2 ' /
'NPUNCH I **Channel for card output (or equivalent) 1.2 ' /
'NREC I **Current record number 1.2 ' /
'NRUN I **Maximum number of steps 1.2 ' /
'NADUMP IA **Codes for array dumps 1.9 ' /
'NPDUMP IA **Codes for dumping points 1.9 ' /
'NVDUMP IA **Codes for dumping arrays 1.9 ' /
'NLCHED L **.TRUE. if class 0 report-head required 1.9 ' /
'NLHEAD LA **.TRUE. if class 1-9 report-heads required 1.9 ' /
'NLOMT1 LA **Class 1 subprogram selector 1.9 ' /
'NLOMT2 LA **Class 2 subprogram selector 1.9 ' /
'NLOMT3 LA **Class 3 subprogram selector 1.9 ' /
'NLREPT L **.TRUE. if report required 1.9 ' /
'NLGRAF L **display device geometry 7.3 ' F/
SF courant_number
&SF DTMUL=1.0,
NINIT=4,
NRUN=100,
NXPTDD=5,
AMODE=1.,1.,1.
CMODE=1.
NMODE=1,1,1/
PS tsd ptflds points times
FS ptflds E1 E2 E3 H1 H2 H3
DS points bitb1
SU bitb1 b1 1 2 3 1. 1. 1. 1. 1.
TS times online
PS tsd avfld face times2
FS avfld intD.dS intB.dS intj.dS
DS face onb1 onb2 onb3 onb4
SU onb1 b1 1 2 3 1. 0. 0. 1. 1. 1.
SU onb2 b2 1 2 3 1. 0. 0. 1. 1. 1.
SU onb3 b3 1 2 3 0. 0. 0. 0. 1. 1.
SU onb4 b4 1 2 3 0. 0. 0. 0. 1. 1.
PS tsd energy volume times
FS energy intE.D/2dV intB.H/2dV intj.EdV
DS volume allb1 allb2 allb3 allb4
SU allb1 b1 1 2 3
SU allb2 b2 1 2 3
SU allb3 b3 1 2 3
SU allb4 b4 1 2 3
TS times2 online 0. 10. 0.2 0. 0. 9
BG type1 regular_cubic_lattice -3 -4 -5
&CUBREG RXMAX=1.0,RYMAX=1.0,RZMAX=1.0/
BG type2 regular_cubic_lattice -3 -4 -5
&CUBREG RXMAX=1.0,RYMAX=1.0,RZMAX=1.0/
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
&UNIFRM /
BL b1 type1 uniform1
BL b2 type2 uniform1
BL b3 sameas b1
BL b4 sameas b1
BL b5 sameas b1
BL b6 sameas b1
BL b7 sameas b1
BL b8 sameas b1
PP pcond perfect_conductor
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=0.5 /
PP aplfld applied_field
&BCS DAPLYA(1)=115.5712,DAPLYA(2)=0.,DAPLYA(3)=0. /
PA pcond sameas pcond
PA reswal sameas reswal
```

95/09/13
08:11:33

une51.uif

2

```
PA aplfld sameas aplfld
BC b1
D=b5:N(1,0)(0,1)
S=b4:D
W=b2:N(1,0)(0,1)
E=b2:S(1,0)(0,1)
N=b4:U
U=b5:S(1,0)(0,1)
EN
BC b2
D=b6:U(0,1)(1,0)
S(1,0)(0,1)=b1:E
W=b3:S(0,1)(1,0)
E=b3:N(0,1)(1,0)
N(1,0)(0,1)=b1:W
U=b6:D(0,1)(1,0)
EN
BC b3
D=b4:W(0,1)(1,0)
S(0,1)(1,0)=b2:W
W=b7:E
E=b7:W
N(0,1)(1,0)=b2:E
U=b4:E(0,1)(1,0)
EN
BC b4
D=b1:S
S=b8:S(1,0)(0,1)
W(0,1)(1,0)=b3:D
E(0,1)(1,0)=b3:U
N=b8:N(1,0)(0,1)
U=b1:N
EN
BC b5
D=b8:D(0,1)(1,0)
S(1,0)(0,1)=b1:U
W=b6:W
E=b6:E
N(1,0)(0,1)=b1:D
U=b8:U(0,1)(1,0)
EN
BC b6
D(0,1)(1,0)=b2:U
S=b7:N(0,1)(1,0)
W=b5:W
E=b5:E
N=b7:S(0,1)(1,0)
U(0,1)(1,0)=b2:D
EN
BC b7
D=b8:E(1,1)(0,0)
S(0,1)(1,0)=b6:N
W=b3:E
E=b3:W
N(0,1)(1,0)=b6:S
U=b8:W(1,1)(0,0)
EN
BC b8
D(0,1)(1,0)=b5:D
S(1,0)(0,1)=b4:S
W(1,1)(0,0)=b7:U
E(1,1)(0,0)=b7:D
N(1,0)(0,1)=b4:N
U(0,1)(1,0)=b5:U
EN
OR b1 b2 b3 b4 b5 b6 b7 b8
```

B PIC3D test data example


```
pc11r01.out      'NLRES' L' /
pc11r01
NI0011 2 turn cold loopback test
```

[illegible]

[illegible]

pc11r01.dat

[illegible]

pc11r01.dat

[illegible]

.....

[illegible]

pci11r01.dat

[illegible]

[illegible]

pc11r01.dat

[illegible]

27:06:42

[illegible]

[illegible]

pc11r01.dat

1	2	3	3/	1	2	3	4	5	10	17/	1	6	13	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	----	---	---	---	---	---	----	-----	---	---	----	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

pc11r01.dat

6	23	7	23	6	7	23	3	9	1	1	14710	0	1	12
6	23	7	23	6	7	23	300	0	0	0	0	0	1	12
10	23	23	23	10	23	23	300	10	0	1	22210	0	1	12
0	23	0	23	0	23	0	300	0	0	0	0	0	1	12
5	23	0	23	5	23	0	2	10	15	1	10	0	1	12
0	23	0	23	0	23	0	312	7200	0	0	0	0	1	12
0	23	0	23	0	23	0	312	7200	15	1	7510	0	1	12
5	23	0	23	5	23	0	312	7200	0	0	0	0	1	12
3	23	0	23	3	23	0	312	7200	15	1	15022	0	1	12
10	23/	23/	23/	10	23/	23/	312	7200	0	1	0	0	1	12
'NXTPSU	IA'						2	7200	0	1	0	0	1	12
20	0	0	0	24	25	26	2	10	1	1	4690	0	1	12
28	0	0	0	32	33	34/	0	7200	0	0	0	0	1	12
'AMODE	RA'						3	9	1	1	12190	0	1	12
39.278538				31.464779	/		0	7200	0	0	0	0	1	12
'BUNI	RA'						3	9	1	1	19702	0	1	12
0.							3	9	1	1	0	0	1	12
'CMODE	RA'			0.	0.15596205E-02/		0	7200	0	0	0	0	1	12
0.83850566E-02/							2	9	1	1	5002	0	1	12
'RPHSHF	RA'						0	7200	0	0	0	0	1	12
'NBEXT	I'						3	8	1	1	12502	0	1	12
'NINIT	I'						0	7200	0	0	0	0	1	12
'NMODE	IA'						3	8	1	1	20014	0	1	12
'NPINIT	I'						2	8	1	1	5314	0	1	12
'NSURBC	I'						0	7200	0	0	0	0	1	12
'ENVCO	RA'						3	7	1	1	12814	0	1	12
'FRQCO	RA'						0	7200	0	0	0	0	1	12
'PHACO	RA'						3	7	1	1	20326	0	1	12
'TINTCO	RA'						0	7200	0	0	0	0	1	12
'MCOSIG	IA'						2	7	1	1	5626	0	1	12
1	1/						0	7200	0	0	0	0	1	12
'MINTCO	IA'						3	6	1	1	13126	0	1	12
1	1/						0	7200	0	0	0	0	1	12
'MTYPCO	IA'						3	6	1	1	20638	0	1	12
5	0	0	0	0	0	0	0	7200	0	0	0	0	1	12
0	0	0	0				2	6	1	1	5938	0	1	12
5/							0	7200	0	0	0	0	1	12
'NCOM	I'						3	5	1	1	13438	0	1	12
'NCOSIG	IA'						0	7200	0	0	0	0	1	12
1	0	0	0	0	0	0	3	5	1	1	20950	0	1	12
0	0	0	0				0	7200	0	0	0	0	1	12
2/							2	5	1	1	6250	0	1	12
'NINSIG	IA'						0	7200	0	0	0	0	1	12
1	2	3	3	4	5	6	3	4	1	1	13750	0	1	12
9	10/						0	7200	0	0	0	0	1	12
'NSIG	I'						3	4	1	1	21262	0	1	12
'EMEATR	RA'						0	7200	0	0	0	0	1	12
'LOREDG	IA'						2	4	1	1	6562	0	1	12
1	63	119	136	139	201	257	0	7200	0	0	0	0	1	12
'LORMDI	IA'						3	3	1	1	14062	0	1	12
1	1	1	1	1	1	1	0	7200	0	0	0	0	1	12
'LORPDI	IA'						3	3	1	1	21574	0	1	12
1	5	7	10	12	16	18	0	7200	0	0	0	0	1	12
'LSTEMB	IA'						2	3	1	1	6874	0	1	12
2	10	1	1	7210	1	1	0	7200	0	0	0	0	1	12
300	0	0	0	0	0	0	3	2	1	1	14374	0	1	12

pc11r01.dat

0	1728	0	1	1	0	0	0	0	1	1	8	1	0	1	5257	1	0	1	1	8
1	1728	0	1	0	1	2361	0	0	0	0	1728	0	0	0	0	1	1	0	1	80
0	1728	0	1	1	1	4169	0	0	1	1	8	1	0	1	1081	1	0	1	1	80
1	1728	0	1	0	0	0	1	0	0	0	1728	0	0	0	0	1	0	0	1	80
0	1728	0	1	1	1	641	0	0	1	1	8	1	1	1	2953	1	0	1	1	80
0	1728	0	1	0	0	0	1	0	1	1	9	1	1	1	4681	1	1	1	1	80
1	1728	0	1	1	1	2441	0	0	1	1	8	0	0	0	0	1	0	0	1	80
0	1728	0	1	0	0	0	1	0	0	0	9	1	1	1	1729	0	0	0	1	8
0	1728	0	1	1	1	1153	0	0	1	72	0	0	0	0	0	1	1	1	1	8
0	1728	0	1	0	0	0	1	0	1	72	8	0	0	0	0	1	1	1	1	8
0	1728	0	1	1	1	2953	0	0	1	72	0	0	0	0	0	1	1	1	1	8
1	1728	0	1	0	0	0	1	0	1	72	9	0	0	0	5329	1	1	1	1	8
1	1728	0	1	1	1	4753	0	0	1	72	0	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	9	0	0	0	1	1	1	1	1	80
0	1728	0	1	1	1	1225	0	0	1	72	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	72	8	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	3025	0	0	1	72	1728	0	0	0	0	0	0	0	0	80
0	1728	0	1	0	0	0	0	0	1	72	1728	0	0	0	0	0	0	0	0	80
1	1728	0	1	1	1	4825	0	0	1	72	1728	0	0	0	0	0	0	0	0	80
0	1728	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	1297	0	0	1	80	1728	0	0	0	9	1	1	1	1	8
0	1728	0	1	0	0	0	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	0	0	0	1	1	1	0	0	0	1801	1	1	1	1	8
1	1728	0	1	1	1	3097	0	0	1	80	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	1728	0	0	0	3609	1	1	1	1	8
1	1728	0	1	1	1	4897	0	0	1	80	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	1369	0	0	1	0	1728	0	0	0	569	1	1	1	1	8
0	1728	0	1	0	0	0	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	1728	0	0	0	2361	1	1	1	1	8
1	1728	0	1	1	1	3169	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	0	0	0	1	1	1728	0	0	0	4169	1	1	1	1	8
1	1728	0	1	1	1	4969	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	1	0	0	0	649	1	1	1	1	8
0	1728	0	1	1	1	1441	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	1728	0	0	0	2441	1	1	1	1	8
0	1728	0	1	1	1	3241	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	1728	0	0	0	1657	1	1	1	1	8
0	1728	0	1	1	1	5041	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	1728	0	0	0	3457	1	1	1	1	8
0	1728	0	1	0	0	0	0	0	1	1	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	1513	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	1728	0	0	0	5257	1	1	1	1	8
0	1728	0	1	1	1	3313	0	0	1	0	1728	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	1728	0	0	0	1729	1	1	1	1	8
0	1728	0	1	1	1	5113	0	0	1	72	0	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	72	1	0	0	0	0	0	0	0	0	8
0	1728	0	1	1	1	1585	0	0	1	72	0	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	1	2	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	72	0	0	0	0	5329	1	1	1	1	8
1	1728	0	1	1	1	3385	0	0	1	72	0	0	0	0	0	0	0	0	0	8
0	1728	0	1	0	0	0	0	0	1	0	9	0	0	0	0	0	0	0	0	80
0	1728	0	1	0	0	0	0	0	1	72	1728	0	0	0	505	1	1	1	1	80
1	1728	0	1	1	1	5185	0	0	1	72	0	0	0	0	0	0	0	0	0	80
0	1728	0	1	0	0	0	0	0	1	72	1728	0	0	0	2377	1	1	1	1	80
0	1728	0	1	1	1	0	0	0	1	72	1728	0	0	0	0	0	0	0	0	80
0	1728	0	1	0	0	1657	0	0	1	72	1728	0	0	0	4105	1	1	1	1	80
0	1728	0	1	0	0	0	0	0	1	72	1728	0	0	0	0	0	0	0	0	80
1	1728	0	1	1	1	3457	0	0	1	2	10	0	0	0	7210	1	1	1	1	12
0	1728	0	1	0	0	0	0	0	1	0	1728	0	0	0	0	0	0	0	0	12

1

[illegible]

pc11r01.dat

0	7200	0	1	0	0	0	0	0	1	12	0	2	1	1	0	0	561	1	1	8
2	8	1	1	1	6610	1	0	0	1	12	0	1728	0	0	1	0	0	0	0	8
0	7200	0	0	0	0	0	0	0	0	12	1	1	1	1	1	1	2361	1	1	8
3	7	1	1	1	14110	1	1	1	0	12	0	1728	0	0	0	0	0	0	0	8
0	7200	0	0	0	0	0	0	0	0	12	1	1	1	1	1	1	4169	1	1	8
3	8	1	1	1	21610	1	1	1	0	12	0	1728	0	0	0	0	0	0	0	8
0	7200	0	0	0	0	0	0	0	0	12	0	1	1	1	1	1	641	1	1	8
2	9	1	1	1	6910	1	1	1	0	12	0	1728	0	0	0	0	0	0	0	8
0	7200	0	0	0	0	0	0	0	0	12	1	0	0	1	1	1	2441	1	1	8
3	8	1	1	1	14410	1	1	1	0	12	0	1728	0	0	0	0	0	0	0	8
0	7200	0	0	0	0	0	0	0	0	12	0	1	1	1	1	1	1153	1	1	8
3	9	1	1	1	21910	1	1	1	0	12	0	1728	0	0	0	0	0	0	0	8
0	7200	0	0	0	0	0	0	0	0	12	1	0	0	1	1	1	2953	1	1	8
0	9	1	1	1	1	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	1	1	1	1	1	1	4753	1	1	8
1	8	1	1	1	1801	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	2	1	1	1	1	1225	1	1	8
1	8	1	1	1	3609	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	1	1	1	1	1	1	3025	1	1	8
0	8	1	1	1	81	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	1	2	1	1	1	1	4825	1	1	8
1	7	1	1	1	1881	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	3	1	1	1	1	1297	1	1	8
1	7	1	1	1	3689	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	1	2	1	1	1	1	3097	1	1	8
0	7	1	1	1	161	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	3	1	1	1	1	4897	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	4	1	1	1	1	1369	1	1	8
1	6	1	1	1	1961	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	3	1	1	1	1	3169	1	1	8
1	6	1	1	1	3769	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	1	3	1	1	1	1	4969	1	1	8
0	6	1	1	1	241	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	4	1	1	1	1	1441	1	1	8
1	5	1	1	1	2041	1	1	1	0	8	0	5	1	1	1	1	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	1728	0	0	0	0	0	0	0	8
1	5	1	1	1	3849	1	1	1	0	8	0	4	1	1	1	1	3241	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	1728	0	0	0	0	0	0	0	8
0	5	1	1	1	321	1	1	1	0	8	0	4	1	1	1	1	5041	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	5	1	1	1	1	1513	1	1	8
1	4	1	1	1	2121	1	1	1	0	8	0	6	1	1	1	1	3313	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	5	1	1	1	1	0	0	0	8
1	4	1	1	1	3929	1	1	1	0	8	0	6	1	1	1	1	5113	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	6	1	1	1	1	0	0	0	8
0	4	1	1	1	401	1	1	1	0	8	0	7	1	1	1	1	1585	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	7	1	1	1	1	0	0	0	8
1	3	1	1	1	2201	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	1728	0	0	0	0	0	0	0	8
1	3	1	1	1	4009	1	1	1	0	8	0	6	1	1	1	1	3385	1	1	8
0	1728	0	0	0	0	0	0	0	0	8	0	7	1	1	1	1	5185	1	1	8
0	3	1	1	1	481	1	1	1	0	8	0	1728	0	0	0	0	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	7	1	1	1	1	1657	1	1	8
1	2	1	1	1	2281	1	1	1	0	8	0	8	1	1	1	1	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	0	8	1	1	1	1	3457	1	1	8
0	2	1	1	1	4089	1	1	1	0	8	0	7	1	1	1	1	0	0	0	8
0	1728	0	0	0	0	0	0	0	0	8	1	1	1	1	1	1	0	0	0	8

pc11r01.dat

[illegible]

'RENORM	R	1	2	3	4	5	6	7	8	0.89860255	1.0269743	1.1553463	1.2837179
'WTDEPC	I	9	10	11	12	13	14	15	16	0.67395192	0.14423393	0.57693571	0.23077428
'RSPOR	RA	17	18	19	20					0.14835489	0.110989252	0.87414503E-01	0.72621278E-01
'LOCPOR	IA									0.62131543E-01	0.54299831E-01	0.48226822E-01	0.43378629E-01
'NINPOR	IA									0.20604849E-01	0.	0.	0.
										2.4973021	0.99892080	0.64216334	0.47567654
'NIPARS	I									0.37837908	0.31434572	0.26894024	0.23504020
'NSPOR	IA									0.20875280	0.18767609	0.89189366E-01	0.64185895E-01
'SSPAR	RA									0.19255771	0.32092953	0.44930127	0.57767314
'TIMDPO	RA									0.70604491	0.83441669	0.96278840	1.0911603
'TIMSSO	RA									1.2195320	1.3479038	0.	0.86540359
'TOUTNS	R									0.28846785	0.17308071	0.12362909	0.96155949E-01
'WINXO	R									0.78673050E-01	0.66569507E-01	0.57693575E-01	0.50906088E-01
'WINXX	R									0.45547560E-01	0.41209698E-01	0.	0.
'WINYO	R									0.86047925E-01	0.78806877E-01	0.72689891E-01	0.67454129E-01
'WINYX	R									0.62921941E-01	0.58960441E-01	0.55468209E-01	0.
'XPSMAX	R									0.17209585	0.15761375	0.14537978	0.13490826
'XPSMIN	R									0.12584388	0.11792088	0.11093642	0.
'YPSMAX	R									0.90396114E-01	0.83379582E-01	0.77373855E-01	0.77373855E-01
'YPSMIN	R									0.72175175E-01	0.67631096E-01	0.63625306E-01	0.
'LMDP	IA									0.25308212E-01	0.23178488E-01	0.21379381E-01	0.19839449E-01
'TIMDOP	I									0.18506452E-01	0.17341305E-01	0.16314179E-01	0.
'NDPO	IA									0.98702028E-01	0.90396121E-01	0.83379589E-01	0.77373855E-01
'NDVECT	I									0.72175168E-01	0.67631096E-01	0.63625298E-01	0.
'NGMAX	I									0.17209585	0.15761375	0.14537978	0.13490826
'NOFSEL	I									0.12584388	0.11792088	0.11093642	0.
'NSI	I									0.17209585	0.15761375	0.14537978	0.13490826
'NSO	IA									0.12584388	0.11792088	0.11093642	0.
'NXPTDD	I									0.86047925E-01	0.78806877E-01	0.72689891E-01	0.67454129E-01
'ECOV	RA									0.62921941E-01	0.58960441E-01	0.55468209E-01	0.
										0.69855654	1.4612988	1.5896708	1.7180427
										1.8464144	1.9747862	2.1031578	1.0836719
										0.69855654	1.4612988	1.5896708	1.7180427
										0.69855654	1.4612988	1.5896708	1.7180427
										1.8464144	1.9747862	2.1031578	1.0836719
										0.10272890	0.21489690	0.23377511	0.25265333
										0.27153152	0.29040974	0.30928791	0.15936352
										0.10272890	0.21489690	0.23377511	0.25265333
										0.27153152	0.29040974	0.30928791	0.15936352
										0.69855654	1.4612988	1.5896708	1.7180427
										1.8464144	1.9747862	2.1031578	1.0836719
										0.69855654	1.4612988	1.5896708	1.7180427
										0.69855654	1.4612988	1.5896708	1.7180427
										1.8464144	1.9747862	2.1031578	1.0836719
										0.69855654	1.4612988	1.5896708	1.7180427
										1.8464144	1.9747862	2.1031578	1.0836719
										0.	0.	0.	0.
										0.	0.	0.	0.
										0.88563561E-02	0.16967436E-01	0.15592583E-01	0.14424118E-01
										0.13418765E-01	0.12544563E-01	0.11777398E-01	0.57089832E-02
										0.17712712E-01	0.33934873E-01	0.31185165E-01	0.28848236E-01
										0.26837530E-01	0.25089126E-01	0.23554796E-01	0.11417966E-01
										0.17712712E-01	0.33934873E-01	0.31185165E-01	0.28848236E-01

C AEA/TYKB/28006/TN/3

AEA/TYKB/28006/TN/3

3DPIC RELEASE 2.10.00: USER GUIDE

J W Eastwood, W Arter, N J Brealey

September 1995

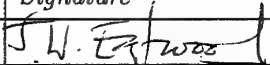
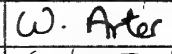
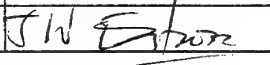
AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB

email: jim.eastwood@aeat.co.uk

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data clause at DFARS 252227-7013.

AEA Technology, Culham Laboratory, Abingdon, Oxfordshire, OX14 3DB, England.

Document Control Number: AEA/TYKB/28006/TN/3			
Date of Issue: September 1995			Issue number: 1
<i>Authorization</i>	<i>Name</i>	<i>Signature</i>	<i>Position</i>
Prepared by	J W Eastwood		Project Manager
Checked by	W Arter		Project Scientist
Approved by	J W Eastwood		Project Manager

Contents

1	Introduction	3
2	Preprocessing using PEGGIE	4
2.1	Input file ml407.uif	5
3	PIC3D on the Workstation	11
4	PIC3D on the SP2	12
5	Using MPICTIM	12
6	Using DISPAN	13
7	Viewing Graphics using GHOST	21

3DPIC RELEASE 2.10.00: USER GUIDE

J W Eastwood, W Arter, N J Brealey

September 1995

Abstract

This Note contains provides an introduction to the user input files (`.uif`), and running the preprocessor PEGGIE the simulation code PIC3D postprocessors TSDMERGE, MPICTIM and DISPAN and the GHOST graphical system.

1 Introduction

Performing a simulation using the 3DPIC software suite is a three stage process:

1. Prepare a user input (`*.uif`) file and run the PEGGIE preprocessor to generate the input data (`*.dat`) file for the main simulation program PIC3D.
2. Run the main simulation program PIC3D to generate the output files

- `*.res` restart file for extending a PIC3D simulation
- `*.out` text output (mainly diagnostic) data
- `*.tsd` timeseries input dataset for MPICTIM
- `*.pnt` ASCII version of `*.tsd` file
- `*.grd` GHOST grid file of initial data
- `*.gr1` GHOST grid file of line graph snapshots
- `*.lin` ASCII version of `*.gr1` file
- `*.gst` GHOST grid file of scatter plot snapshots
- `*.pts` ASCII version of `*.gst` file

If PIC3D is run on a several processors, then each process generates an output file. The GHOST grid file outputs can be combined by superposing them using the interactive GHOST postprocessor `xghost`. The timeseries (`*.tsd`) files from each process are combined using the postprocessor TSDMERGE before exporting the files back to the workstation on which MPICTIM is run.

3. Run the postprocessors (MPICTIM, DISPAN, GHOST) to analyse the output files.

The following Sections give examples of running the various components of the 3DPIC software suite.

2 Preprocessing using PEGGIE

To run the preprocessor PEGGIE and generate the data file for the main simulation run issue the `peggie` command with the name of the user input file as the argument. For example,

```
% peggie ml407.uif
```

This generates the output file `ml407.dat` for use by the main simulation code PIC3D and optionally a graphical output file containing a wire grid plot of the element net generated by PEGGIE with colour coding of each block. The present Version of PEGGIE generates a large number of diagnostic warning messages, which can be ignored (but error messages should not be ignored if they appear!), and then goes on to output prompts for the interactive choice of parameters for the grid display:

```

      VPNX      = -1.0000E+00
/
      VPNY      = -3.0000E-01
/
      VPNZ      =  7.0000E-01
/
      VRCX      =  0.0000E+00
/
      VRCY      =  0.0000E+00
/
      VRCZ      =  0.0000E+00
/
      VUPX      =  0.0000E+00
/
      VUPY      =  1.0000E+00
/
      VUPZ      =  0.0000E+00
/
      xmin map  = -1.2500E-01
-400
      xmin map CHANGED TO  -4.0000E+02
      xmax map  =  1.2500E-01
400
      xmax map CHANGED TO   4.0000E+02
      ymin map  = -1.2500E-01
-400
      ymin map CHANGED TO  -4.0000E+02
      ymax map  =  1.2500E-01
400
      ymax map CHANGED TO   4.0000E+02

      plotype   =           2

```

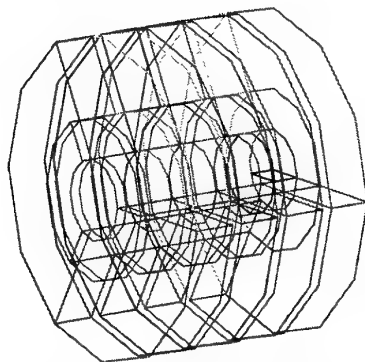


Figure 1: The wire grid plot of block edges generated by the `m1407.uif` example run of PEGGIE.

```
3
  plotype CHANGED TO          3
another projection? y/n
n
```

In the above dialogue, the user reply / to the displayed value leaves that value unchanged. The first three items `VPNX`, `VPNY` and `VPNZ` specify the vector giving the viewing position, the next three (`VRCX`, `VRCY` and `VRCZ`) define the reference origin point on the element net, and the third triplet define the 'up' direction for the plot. The minimum and maximum map values define the extent of the mapping to the viewing plane to be displayed. `plotype` is 1 for all element edges to be plotted, 2 for element edges on block faces to be plotted, and 3 for only block edges to be plotted. The result for this example is shown in Fig 1

2.1 Input file `m1407.uif`

In this Subsection, a step-by-step discussion of a sample `.uif` file is given to illustrate the method of building up geometry, etc using parts. A full definition of the input formats used in the `.uif` files is given in the Note [7], and that document should be referred to when reading the following example.

The example input file used is `m1407.uif` which sets up a four cavity MILO, using a cylindrical block for the diode region and a set of 120 degree angle annular segments for the drift space and cavities. This MILO example, and full `.uif` files for a number of other test cases is presented in Reference [6].

The initial part of the `m1407.uif` file is concerned with labelling; the only important label is the reference labelling string `m1407` which propagates through all the files generated by 3DPIC and should be unique to each simulation experiment to prevent filename clashes with other runs.

The line

```
'NLGRAF  L  **display device geometry          7.3  ' T/
```

is used to select the display of the device geometry. If the `T/` is replaced by `F/`, then the device geometry is not plotted.

set free parameter The tag SF indicates that the SF namelist follows with the various free parameters to be set. In this example, the Courant Number is set to 0.99 (DTMUL), the run length is 2000 steps and diagnostic output is suppressed (NXPTDD). NTPRES is the number of processors on the target machine. The \ terminates the namelist.

```
SF courant number
&SF DTMUL=0.99,
NRUN=2000,
NTPRES=1,
NXPTDD=5/
```

scatter plots The next sequence specifies the scatter plots. The first line (tag PS) defines the plot set

```
PS gst pcles alla times
```

This says that the plot set is a grid file scatter plot type (.gst) for the field set pcles over domain set alla for the time set times.

The following extract defines the field set pcles:

```
FS pcles pplot1 pplot2
FL pplot1
&FL CFLNAM='r(z)',LFSKIP=10,LFCOLO=3,
FXMIN=-0.2,FXMAX=0.05,FMIN=0.,FMAX=0.150/
FL pplot2
&FL CFLNAM='theta(z)',LFSKIP=10,LFCOLO=4,
FXMIN=-0.2,FXMAX=0.05,FMIN=-180.,FMAX=180./
```

This field set (tag FS) contains fields pplot1 and pplot2. Parameters for the field pplot1 are given by the namelist following the FL tag for that field; in this instance it is a $r - z$ scatter plot, plotting every tenth particle in GHOST colour 3 (green) with the abscissa variable (z) over range -0.2 to 0.05 and the ordinate variable over range 0 to 0.15. Similarly pplot2 is a $\theta - z$ scatter plot, this time using blue dots for the particles.

The domain set alla is defined by the DS line; in this case, there is one domain named alldo, comprising sub-domains b1s ... b4Es. The lines tagged by SU link the subdomains to whole blocks in this instance.

```
DS alla alldo
DO alldo b1s b2s b3s b4s ca2s ca3s ca4s b2cs b3cs b4cs ca2cs +
ca3cs ca4cs b2bs b3bs b4bs ca2bs ca3bs ca4bs b2as b3as b4as +
ca2as ca3as ca4as b2Es b3Es b4Es
SU b1s b1
SU b2s b2
SU b3s b3
SU b4s b4
SU ca2s ca2
SU ca3s ca3
```


The time set `times` is defined later in the `.uif` file. Since this is the first plot set defined, it will lead to the output file `m13407-01.gst` being generated when PIC3D is executed on a single processor workstation. On a multiprocessor machine, the process number is also appended to the root of the file name (eg `m13407-01-0002.gst` for process 2 output and so forth).

point field plots The second plot set, whose output will go to the output file `m13407-02.tsd` is a time series data file format, and will contain point field values defined by the field set `ptflds` for the domain set `points` sampled for time set `times2`.

```
PS tsd ptflds points times2
FS ptflds E1 E2 E3 H1 H2 H3
DS points pt1 pt2 pt3 pt4 pt5
SU pt1 b2E 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt2 b2a 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
SU pt3 b2c 1 2 3 0.5 0.5 0.2 0.5 0.5 0.2
SU pt4 b2 1 2 3 0.5 0.5 0.0 0.5 0.5 0.0
SU pt5 b2 1 2 3 0.5 0.5 0.6 0.5 0.5 0.6
```

The sub-domain definitions are a little more involved than for the scatter plot case, with three triplets of numbers after the block name. The first gives the direction for evaluating components – in this case the block (x^1, x^2, x^3) are the (1, 2, 3) components, respectively. The next two triplets the 'origin' point and the last is the 'extreme' point which define the diagnostic volume in block coordinates with components on interval [0,1]. For instance, `pt5` is at point (0.5, 0.5, 0.6).

The next section of the `.uif` file defines a second `.tsd` file, this time for points in the cavity blocks.

```
PS tsd ptflds cavs times2
DS cavs cav2 cav3 cav4 cav5
SU cav2 ca2a 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav3 ca2b 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav4 ca2c 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
SU cav5 ca2 1 2 3 0.8 0.5 0.5 0.8 0.5 0.5
TS times online
TS times2 online 0. 50. 0.006 0. 0. 9
```

The last two lines of this fragment define the time sets `times` and `times2`. The time set `times` specifies that data be collected at default 'sensible' times. The `times2` time set specifies that data be collected between times 0 and 50 at intervals .006, where the 9 indicates that time units are nanoseconds.

line graphs The section defining the line graph follows the same pattern as the other diagnostics, with definitions of plot set, field set and domain set. The domain outer is a curve, and the tag CU introduces the namelist INHERI, which defines the start point (0, 0, 0.8) on block b2b for a curve which propagates along element edges in the 2-direction through surface patches to neighbouring blocks until it closes or terminates at a boundary surface. In this case, the curve is a closed circular curve enclosing the cathode.

```
PS gr1 Efield outline times
FS Efield E1 E2 E3
DS outline outer
CU outer inherit
&INHERI CBLOCK='b2b',
OPOINT=0.,0.,0.8,
NDIR(1)=2,
NLXTEN=T,CUNIVS='all'/
```

surface integrals The final plot set generates a .tsd file which gives radial fluxes through a surface in block b2.

```
PS tsd avfld face times2
FS avfld intD.dS intB.dS intj.dS
DS face onb2
SU onb2 b2 1 2 3 0.4 0. 0. 0.4 1. 1.
```

block types The above completes the specification of the diagnostics. The next stage is to define the geometry and physics of the parts which make up the block types used in describing a device. Tag BG flags block geometry and BP flags block physics.

```
BG centre polar_with_regular_meshing -8 -12 -10
&POLREG RADINR=0.,RADOUT=0.0375,THEMAX=360.,AXMAX=0.025/
BG outer polar_with_regular_meshing -8 -4 -16
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.04/
BG cavity polar_with_regular_meshing -16 -4 -16
&POLREG RADINR=0.075,RADOUT=0.150,THEMAX=120.,AXMAX=0.04/
BG inout polar_with_regular_meshing -8 -4 -20
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.05/
BG first polar_with_regular_meshing -8 -4 -4
&POLREG RADINR=0.0375,RADOUT=0.075,THEMAX=120.,AXMAX=0.01/
BP uniform1 uniform
&UNIFRM EPSR=1.,RMUR=1./
BP uniform2 uniform
```

All the BG parts defined above are of the class `polar_with_regular_meshing`. The first part (`centre`) has a suggested mesh of $8 \times 12 \times 10$ elements, and forms a complete cylinder ($\theta_{max} = 360$) with outer radius 0.0375 m, axial

length of 0.025 *m*. Both block physics parts are of class **uniform** with relative permittivity and permeability of unity.

blocks Each block has a label, a geometry part and a physics part. For example, block **b2** has an **outer** block geometry and a **uniform1** block physics. Block **b3** has the same block geometry and physics and block **b2** and so forth.

```
BL b1 centre uniform1
BL b2 outer uniform1
BL b3 sameas b2
BL b4 sameas b2
BL ca2 cavity uniform1
BL ca3 sameas ca2
BL ca4 sameas ca2
BL b2a inout uniform1
.
.
.
```

surface patches The whole of the exterior surface of the device must be covered by surface patches, and each of these patches have associated patch physics. The patch physics tags (PP) and associated boundary condition namelists below introduce perfect conductors (**pcond**), a 377 Ω per square resistive surface (**reswal**), a radial electric field corresponding to an applied voltage of amplitude 0.5 MV with E_r radially inwards, and a polar axis. The namelist quantity **SFORM** specifies the signal waveform part – in this case **boundary** (cf applied signal paragraph below). The PA tags then associate these electromagnetic patch physics conditions with patch names.

```
PP pcond perfect_conductor
&BCS /
PP reswal resistive_wall
&BCS SURFZ=377.,STHETA=1.0 /
PP aplfld applied_field
&BCS DAPLYA(1)=-1.,DAPLYA(2)=0.,DAPLYA(3)=0.,
DPOT=0.5E+6,SFORM='boundary'/
PP axis polar_axis
&BCS /
PA pcond sameas pcond
PA reswal sameas reswal
PA aplfld sameas aplfld
PA axis sameas axis
```

If charged particle source boundary conditions are associated with a patch, then the patch definition requires both particle and electromagnetic patch physics. The following species definition (tag **SP**) defines species **electrons** of class **any** of mass m_e , charge $-|e|$ with 10^9 particles per superparticle, a nominal 8 particles per element and a subcycling of 1 electromagnetic timesteps

per particle timestep. The PP tag defines part **emitter** to be of class **emitter** where the class **emitter** namelist **PARBCS** allow particle boundary condition parameters to be set. The patch definition tag **PA** associates **emitter** particle patch physics and **pcond** electromagnetic particle boundary conditions with patch part **pemcon**.

```
SP electrons any 1 -1 6 9 8 1
PP emitter emitter
&PARBCS CSPECI='electrons',EFFE=1.0/
PA pemcon emitter pcond
```

applied signals The signal part boundary has component parts **ramp** and **signalcpt**; **ramp** applies a linear ramp up from 0 to the signal voltage (**DPOT**) over an interval of 3 nanoseconds, and **signalcpt** hold the voltage constant over the next 10,000 seconds.

```
SG boundary ramp signalcpt
SC ramp
&SIGNAL
  FREQUENCY = 0.0,
  DURATION = 0.3E-9,
  POWER = 0.0, 1.0,
  SIGNAL_FORM='LINEAR'/
SC signalcpt
&SIGNAL
  DURATION = 10000.,
  POWER = 1.0, 1.0,
  SIGNAL_FORM='CONSTANT'/
```

block connections The final task is to glue all the blocks together by specifying the block connectivity for each block. There is some redundancy in this specification, but this is checked by **PEGGIE** in its consistency checks when building the patch tables.

```
BC b2
W(0,0)(1,0.375)=pemcon
W(0,0.375)(1,1)= b1:E (0, 0)(0.333333,1)
N=b3:S
S=b4:N
E=ca2:W
D=b2c:U
Up =pcond
EN
.
.
BC ca2
W=b2:E
N=ca3:S
```

```

S=ca4:N
E=pcond
D=pcond
U=pcond
EN
.
.

```

Each block has a north, south east, west, up and down face, as described in [7]. For full face connection with normal orientation, one simply has to specify which face of which target block the current block is being connected. Thus, for cavity block **ca2**, the west face is connected to the east face of block **b2**, the north face is connected to the south of block **ca3**, the south face is connected to the north face of block **ca4**, and the east, down and up faces are connected to perfect conductors.

For part-face connections, the local block coordinates of the 'O' and 'X' corners of the connected parts of the face also need specifying. Thus, for the 120 degree annular wedge block **b2**, the west face (ie the surface at **RADINR**) is partly covered by the cathode emitting conductor surface **pemcon** (for x^3 from 0 to 0.375) and partly connected (for x^3 from 0.375 to 1) to the east face (between $\theta = 0$ and $\theta = 120$ degrees) of block **b1**.

Finally, the order in which the block connection (tag **OR**) can be specified to determine which block meshing density takes priority if inconsistencies are detected.

```

OR b1 b2 b3 b4 ca2 ca3 ca4 b2c b3c b4c ca2c ca3c ca4c b2b b3b
b4b ca2b ca3b ca4b b2a b3a b4a ca2a ca3a ca4a b2E b3E b4E

```

This is the end of the user input file **m1407.uif**. Further examples may be found in Reference [7]

3 PIC3D on the Workstation

To run the simulation program **PIC3D** issue the command **pic3d** with the name of the data file as an argument. For example,

```
% pic3d m1407.dat
```

The selection of output files and their contents depends on the specification given in the **.uif** file. In this instance, the following files were generated:

m1407.out	text output (mainly diagnostic) data
m1407.grd	grid file containing labelling graphics page
m1407-01.gst	grid file containing scatter plots
m1407-02.tsd	time series data input for MPICTIM
m1407-03.tsd	time series data input for MPICTIM
m1407-04.gr1	grid file containing line plots
m1407-05.tsd	time series data input for MPICTIM

4 PIC3D on the SP2

The main differences between running PIC3D on a workstation and in multiprocessor runs on the SP2 are that the input file has a specific name `input.dat`, and that the execution of PIC3D leads to separate output files for each processor.

The procedure for executing PIC3D on the SP2 is

1. import the `.dat` file to the file `input.dat`,
2. set up a PVM3 parallel machine on the SP2,
3. ensure left-over files from earlier runs have been deleted,
4. execute `pic3d`
5. merge output `.tsd` files,
6. export output files back to the workstation for postprocessing.

These steps are described in detail in the "readme" file

`~hockney/MAUI-3DPIC/README-MIMD`

on the MHPCC SP2 computer. A copy of this "readme" file is attached as Annex G of the Report [4]

5 Using MPICTIM

The version of MPICTIM (see Technical Note [8]) provided with the current release of 3DPIC differs from the earlier version provided in that it allows graticules on graphs, logarithmic scales, and point-and-click extraction of data (x, y) values from plots.

The procedure for getting displays is the same as for earlier version. The basic steps are to

1. execute `mpictim` to start the Motif dialogue panel and graph panel,
2. open the data file,
3. select the plot type,
4. set plot ranges,
5. plot curves and/or axes,
6. save and/or erase plot.

For further details, refer to Technical Note [8].

6 Using DISPAN

The dispersion analysis postprocessor DISPAN allows the interactive interrogation of the .lin datasets generated by 3DPIC. The mathematical description of the analysis performed by DISPAN is described in Reference [6].

The example given below illustrates how DISPAN may be used to extract frequency and wavenumber information from datasets generated by PIC3D.

DISPAN takes the user through the following sequence of operations:

1. file selection
2. domain selection
3. field selection
4. spatial fourier transformation
5. selection of time interval
6. time FFT
7. structure factor display
8. frequency spectra at given k
9. further selection options

These are illustrated below using extracts from an interactive session using DISPAN on a dataset pc11r02-02.lin generated by PIC3D from a 'cold-test' run (ie a purely electromagnetic calculation) on a five turn section of an helix TWT. It is assumed that the data files vtube1.dat and pc11r02-02.lin are in the present working directory (pwd) and that dispn is on the search path.

The session begins by typing:

```
dispan vtube1.dat
```

The data file vtube1.dat contains the following

```
tube1.odp
'NLRES L False for NEWRUN,True for RESET' /
tube1
DISPAN\ Version 1.00.00 January 1995
Dispersion analysis tool
For use on .lin output files from PIC3D Version 2.10.00
Defaults below are for Tube 1 data
$NEWRUN
    FRQMIN= 0.0,
    FRQMAX= 35.0,
    BETAMN=-4488
    BETAMX= 4488
    NOCHTD=25,
```

```

MCHMIN=10,
MCHMAX=25
$END

```

The first line gives the name of the output log file for the session; in this case it is `tube1.odp`. The second and third line are respectively the restart flag (dummy in this instance) and reference label `tube1`. The next four lines are labelling for the output log.

The main purpose of the data file `vtube1.dat` is to provide default ranges for wavenumbers and frequencies on the plots. Choosing appropriate values avoids a lot of resetting of plot ranges during the interactive session. In this cases, we are looking at data from a helix with pitch $p = 1.410^{-3}m$, and are interested in data in the first zone, so a wavenumber range $[-K, K]$, where $K = 2\pi/p = 4488m^{-1}$ and frequency range $[0, 35 \text{ GHz}]$ make reasonable choices. Wavelength ranges are set by parameters `BETAMN`, `BETAMX`, and frequency ranges are set by `FRQMIN` and `FRQMAX`. The number of uniformly spaced contour heights spanning the data is given by `NOCHTD`, and the range of contour heights plotted is given by `[MCNMIN, MCHMAX]`.

File Selection. The first interaction is the selection of the `.lin` dataset. The program prompts for the root of the file name, and asks for verification if it finds it in the `pwd`.

```

Please enter the root of the .lin file name
pc11r02-02
Open data file? (y|n)[y] name: pc11r02-02.lin
y
Grid file called pc11r02.gdp

```

If the data is not found, the program will repeat prompting until a valid file is found. If it is found, then `DISPAN` announces the name it generates for the output grid file (`pc11r02.gdp` in this example).

Domain Selection. A dataset may contain line data along domains, then a menu is displayed and the users selects the one he wants. In this example, only one domain is present and so it is automatically selected.

```
DOMAIN SEARCH COMPLETED
```

```
Domain Selection Menu
```

```

only one option available
Option selected: axial distance (m)

```

Field Selection. Each domain may contain data for several fields. For example, a domain could be a line along the axis of the device on which components of magnetic and electric field are accumulated. Here, only one field, the scalar helix to body tube voltage is stored, so again the selection defaults to the only one available.

Field Selection Menu

```

only one option available
Option selected: radial voltage (helix - body tube)
Now loading dataset - please wait
radial voltage (helix - body tube)

```

Spatial Fourier Transform. Once the data is read in, it is spatially fourier transformed. The temporal transform is designed for use with a small number of periods of the fundamental signal ($\sim 1-2$), and this necessitates the choice of interval for the transform to fit the period of the data to get good spectral results. The alternative, using a cosine bell and zero padding to eliminate spectral aliasing was not adopted as it requires a much larger number of periods of oscillation in the data to get good spectral resolution. The method for selecting the data time interval implemented is a visual point-and-click approach, as described below.

Selection of Time Interval. The following extract plots the time evolution of the $k = 1$ harmonic (ie wavenumber $\beta = 2\pi k/L = 2\pi/L$, where L is the domain arc length = $5p$ for this example). Where yes/no answers show default option (eg [n]), then simply pressing return takes that default.

```

Select wavenumber index from range 0 to 60
1
Set plot ranges? [n]

Time Filtering? [n]

Extract Data Values? [n]
n

```

The result of this step is the plot shown in figure 2. When viewed on the screen, the red curve corresponds to the cosine harmonic, and the blue to the sine harmonic.

It is obvious from figure 2 that the low frequency component is difficult to identify, so we replot the data, this time passing it through a low pass filter to make the lower frequency/longer wavelength component easier to identify. The following extract from the terminal session shows the user select a plot range for amplitude from -30 to 30, and a width 11 for the time filter. The resulting plot is shown in figure 3.

```

Replot same data? [n]
y
Set plot ranges? [n]
y
Harmonic amp min = -268.24      max = 285.23      OK y/n?
n

```

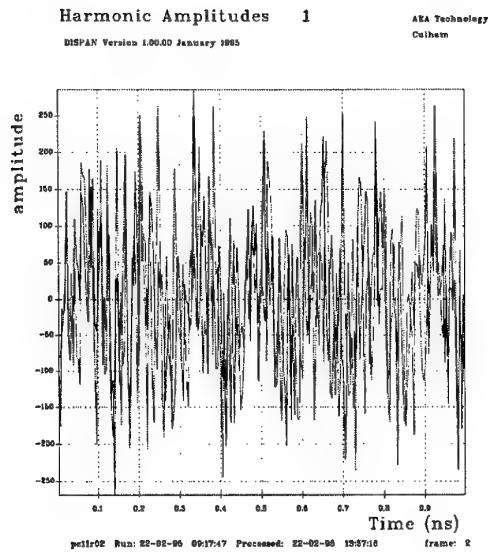


Figure 2: The time evolution of the $k = 1, \beta = 877.6m^{-1}$ harmonic

```

Input new min and max
-30 30
  Harmonic amp min = -30.000    max = 30.000    OK y/n?
y
  Time range min = 0.40045E-02 max = 0.99712    OK y/n?

Time Filtering? [n]
y
Input filter length (1-100)
11

```

The fundamental mode in figure 3 is now easy to identify. Answering y to the Extract Data Values query activates the cursor, and pressing the left mouse button at the cursor position (marked by crosses on figure 3) prints values in the dialogue screen. In this example, six oscillation periods have been chosen. Pressing the right mouse button terminates the cursor interaction, and answering no to the next two prompts brings up the selection of the time range. Here the values obtained at the cursor have been used and confirmed.

```

Extract Data Values? [n]
y
  Mouse left button for data, right for done
  Time = 0.10841    Amplitude = 0.16281E-01
  Time = 0.95892    Amplitude = -0.10914
Replot same data? [n]

Another Harmonic? [n]

```

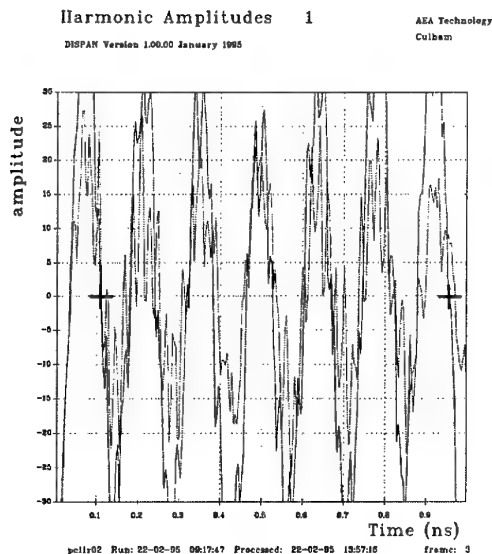


Figure 3: A time filtered plot for the time evolution of the $\ell = 1$ harmonic

```

TIME RANGE min = 0.40045E-02 max = 0.99712      OK y/n?
n
Input new min and max
.1084 .9589
TIME RANGE min = 0.10840      max = 0.95890      OK y/n?
y

```

Time FFT. Once the time range is entered, DISPAN performs a periodic spline interpolation to a set of 2^N uniformly spaced points spanning the time range, and then performs a periodic fourier transform.

Structure Factor Display. The following dialogue shows the bidirectional spectrum selected. The range of values of S are spanned uniformly by the NOCHTD contours (= 25 in this case) and the user is prompted for the subset of these heights to be plotted.

```

bidirectional spectrum? [y]
y
Reset plot range? [n]

Amplitude normalised plot? [n]

```

```

min cont  =      10
/
max cont  =      25
/

```

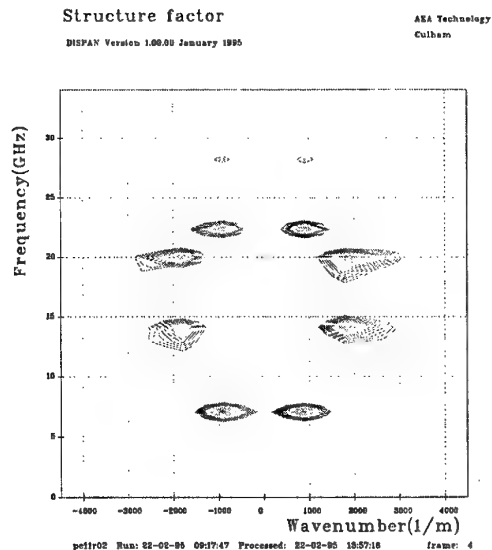


Figure 4: The structure factor plot. The peaks in the contour plots show locations of points on the ω/β diagram

Extract Data Values? [n]

n

The result is plotted in figure 4. It is usually easier to pick out the spectral peaks in structure factor plots where the maximum at each wavenumber is normalised to unity. The next dialogue extract and figure 5 shows this:

Replot same data? [n]

y

Reset plot range? [n]

Amplitude normalised plot? [n]

y

min cont = 10
20
min cont CHANGED TO 20
max cont = 25

/

Extract Data Values? [n]

y

Mouse left button for data, right for done
beta = 885.43 rad/m, f = 7.0337 GHz
beta = 1835.1 rad/m, f = 14.082 GHz
Replot same data? [n]

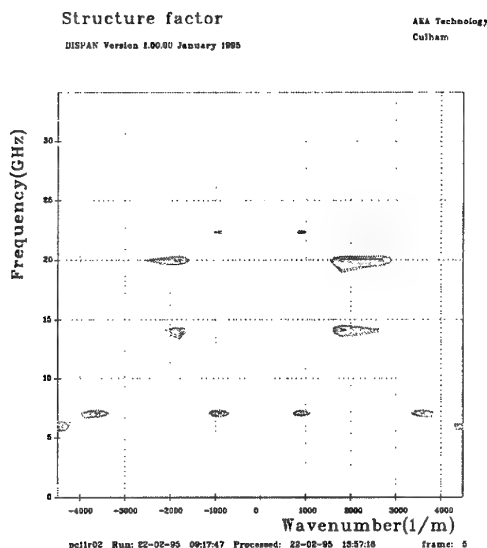


Figure 5: The normalised structure factor plot

The f and β values can be extracted directly from the structure factor plots by clicking the mouse at the cursor position, as illustrated in the above extract, or can be measured more precisely from the spectral plots at given wavenumbers.

Frequency Spectra. After completing the structure factor dialogue, DISPAN moves on to plotting frequency spectra. The following dialogue overplots the spectrum for left going waves ($k = 1$, solid curve) and right going waves ($k = -1$, broken curve) at wavenumbers $\beta = 897.6\text{m}^{-1}$, and then extracts β and f using the cursor.

```
Select wavenumber index from range -60 to 60
1
Set plot ranges? [n]

Extract Values? [n]
n
Replot same data? [n]
n
Another Harmonic? [n]
y
  Select wavenumber index from range -60 to 60
  -1
  Set plot ranges? [n]

Overplot? [n]
y
Extract Values? [n]
y
```

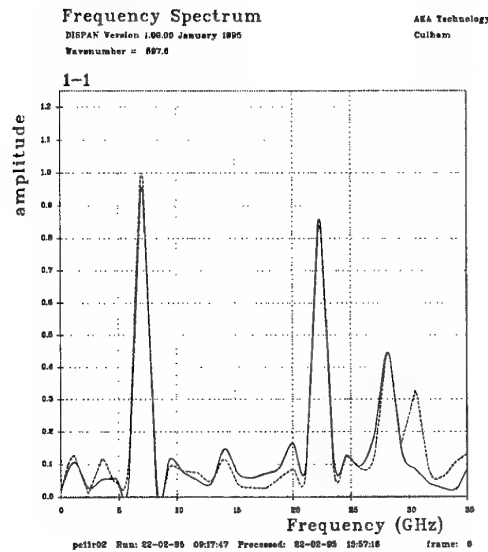


Figure 6: Frequency spectrum plots at $\beta = 897.6m^{-1}$. The solid and dashed curves and respectively for the left and right going waves. The peaks correspond to the fundamental and $\beta = 3590.4m^{-1}$ space harmonic.

```
Click cursor to right of plot to end
Mouse left button for data, right for done
Frequency = 7.0007      Amplitude = 1.0012
Frequency = 22.320     Amplitude = 0.85528
Replot same data? [n]
n
```

The plot corresponding to this dialogue is shown in figure 6. Further wavenumber spectra can be similarly be generated and plotted on the same or separate graphs.

Further Selections. Answering no to the **Another harmonic** query completes the interrogation of the current domain/field selection. For domains with multiple fields and datasets with multiple domains, the dialogue returns to the appropriate menu. In this example, the only further option is another data file. Selecting no completes the session.

```
Another Harmonic? [n]
```

```
Another data file? [y]
```

```
n
```

A further example of the use of the dispersion analysis postprocessor is given in Section 5.5 of Reference [6].

7 Viewing Graphics using GHOST

The simplest GHOST tool for viewing the grid graphical files is the command line interface tool `xghost`. For example, to view the frames in file `m1407-01.gst` sequentially, simply type

```
xghost m1407-01.gst
```

and click on the graphics window to advance the frame.

To select specific frames, and to overplot, scale, rotate, etc, `xghost` must be used in interactive mode:

```
xghost -int
CMND? pict m1407-01.gst 3 5
.
.
.
CMND? gr
```

The GHOST graphical (grid) files can be converted into HPGL2, CGM or POSTSCRIPT (and many other formats) for further processing or printing. For example, to generate monochrome Postscript output in portrait orientation in output file `sample.ps` from the input file `sample.grd`, type

```
postsc -gpor -gfil sample.ps sample.grd
```

For colour Postscript or HPGL2 output, substitute respectively `postcl` or `hpgl2` for `postsc` in the above example.

For more details on `xghost`, `postsc`, etc see the GHOST User Manual [3] section entitled "GHOST for UNIX".

References

- [1] N J Brealey, J W Eastwood and W Arter, *3DPIC Diagnostics*, Technical Note AEA/TYKB/31878/TN/9, Culham Laboratory, September 1994.
- [2] N J Brealey, *MPICIM User's Guide*, Technical Note AEA/TYKB/31878/TN/10, Culham Laboratory, September 1994.
- [3] W A J Prior, *GHOST USER MANUAL Version 8*, UKAEA Culham Laboratory, ISBN 0-85311-184-7, 1991.
- [4] J W Eastwood, W Arter, N J Brealey and R W Hockney, *3D Body-Fitted Software on the MHPCC SP2 Computer*, Report AEA/TYKB/28006/RP/1 Culham Laboratory, September 1995.
- [5] R W Hockney, *LPM3 Benchmark Results on the MHPCC IBM SP2 Computer*, Technical Note AEA/TYKB/28006/TN/1, Culham Laboratory, September 1995.

- [6] J W Eastwood, W Arter, N J Brealey and R W Hockney, *Demonstration Calculations of HPM Sources using 3-D Electromagnetic PIC Software*, Technical Note AEA/TYKB/28006/TN/2, Culham Laboratory, September 1995.
- [7] W Arter, *Release 2.10.00 of the PEGGIE Preprocessor*, Technical Note AEA/TYKB/28006/TN/4, Culham Laboratory, September 1995.
- [8] N J Brealey, *MPICTIM Release 2.10.00: User Guide*, Technical Note AEA/TYKB/28006/TN/5, Culham Laboratory, September 1995.

D AEA/TYKB/28006/TN/4

AEA/TYKB/28006/TN/4

RELEASE 2.10.00 OF THE PEGGIE
PREPROCESSOR

W Arter

September 1995

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB

Document Control Number: AEA/TYKB/28006/TN/4			
Date of Issue: September 1995			Issue number: 1
<i>Authorization</i>	<i>Name</i>	<i>Signature</i>	<i>Position</i>
Prepared by	W Arter	W. Arter	Project Scientist
Checked by	J W Eastwood	J. W. Eastwood	Project Manager
Approved by	J W Eastwood	J. W. Eastwood	Project Manager

Contents

1 INTRODUCTION	3
2 KEY CONCEPTS	4
3 BASIC INPUT FORMATS	6
3.1 Standard Format	6
3.2 List of Labels	7
3.3 Connectivity Format	7
4 DETAILED INPUT SPECIFICATION	8
4.1 Tags in Standard Format	8
4.1.1 BG, BP, CU and PP	8
4.1.2 DE, FL, PO, SC and SF	8
4.1.3 TS and SU	11
4.1.4 SP	12
4.2 List of Labels Tags	12
4.2.1 BL and PA	12
4.2.2 OR	12
4.2.3 PS	12
4.2.4 FS	13
4.2.5 DO	13
4.2.6 DS	13
4.2.7 SD	14
4.2.8 SG	14
5 CLASS DESCRIPTIONS	14
5.1 Block Geometry	14
5.1.1 Rectbk	14
5.1.2 Genblk	14
5.1.3 Parallelepiped	15
5.1.4 Polar with regular meshing	15
5.1.5 Variable spaced polar lattice	15
5.1.6 Variable spaced cubic lattice	15
5.1.7 Polar mesh segment	15
5.1.8 Polar to rectangular transition	16
5.1.9 Regular cubic lattice	16
5.1.10 Quadrilateral cylinder	16
5.2 Block Physics	16
5.2.1 Uniform	16
5.3 Particle Properties	17
5.3.1 Extra and Below	17
5.4 Patch Physics	18
5.4.1 Perfect conductor, Polar axis and Reflector	18
5.4.2 Applied field, Resistive wall and Surface field	18
5.4.3 Emitter, Inject beam and Multipaction	18

5.5	Other Geometry	19
5.5.1	Inherit	19
5.6	Miscellaneous	19
5.6.1	Free parameters	19

RELEASE 2.10.00 OF THE PEGGIE PREPROCESSOR

W Arter

September 1995

Abstract

The concepts underlying our approach to mesh generation are outlined, and then a detailed specification is presented. This document extends earlier notes issued on this topic [3, 4, 5] to include the interface to produce parameters that cause PIC3D to perform simulations of particle beams and secondary emission.

1 INTRODUCTION

This Note describes the inputs acceptable to the PEGGIE (Particle Electromagnetic General Geometry InterfacE) preprocessor for the 3DPIC simulation package. The basic mesh generation function of PEGGIE has now been enhanced and extended to the point where it can produce datasets that will cause PIC3D to simulate complex physics, such as electric fields on resistive walls and particle emission. However only discussion of the grid generation issues appears here. For the additional topics, this publication serves as only a user manual.

The complexity of interfaces to existing mesh generation software has led us to produce a simpler interface tailored to the needs of the program under development. The concept underlying our approach is that the device to be modelled is made by joining together 'parts', with appropriate coding to generate the mesh and associated information for each class of part. One important advantage is that 'appropriate' coding could just be that which converts the output of another mesh package into a format consistent with our new software.

For reasons of simplicity and portability, input to PEGGIE takes the form of a file containing text. Previous experience highlights the value of using tags within the file, that take the form of two character words appearing at the start of an item. The tags serve to tell the processor the format of subsequent data. The following discussion indicates the formats that may be required.

First, let us observe that each class of part must have a unique name since it may correspond to a piece of code, which good programming practice implies must be a subroutine. In this case there will be parameters to be set, e.g. the dimensions of a block and the number of mesh-points. Several different parts may belong to the same class, hence some sort of marking is needed to distinguish one part from another. Moreover, there are effectively two sorts of

class, not only classes which correspond to subprograms of the mesh software, but also those that are determined dynamically e.g. by reading the output of another mesh generator. Such output may be extremely verbose, hence it is desirable to be able to tell the code to look in a disk file for the data.

The characteristics required of the most general tagged record are thus that it should contain a label, a class name, some way of specifying whether a disk file is to be searched, and then some parameters. Experience again suggests that all but very small sets of parameters are best specified using the NAMELIST format of input. NAMELIST formats are defined as part of the FORTRAN 90 standard[2], and most FORTRAN 77 compilers of which we are aware accept NAMELIST, with slight variations that can be easily corrected for. NAMELIST is not so useful if all parameters in a set have to be specified, hence both NAMELIST and ordinary lists will be permitted. The resulting record structure enables compactness of representation coupled with great flexibility and readability.

The key concepts underlying our approach to mesh generation are further expounded in Section 2. Section 3 describes the three basic input formats required, of which the first or standard is by far the most important. Section 4 contains a detailed specification for input to PEGGIE, and the final section 5 describes the classes available. Section 3 also includes a description of the interface to the production of parameters that control output from a PIC3D run.

2 KEY CONCEPTS

As already mentioned, a device is conceived of being made up of parts. Let us think of a part as relating to a single multiblock or uniblock. For book-keeping purposes it is necessary for every uniblock to have a unique label. However, just as say many cars have four identical wheels, so several uniblocks may correspond to the same part. Defining a uniblock thus involves (i) giving the uniblock a label and (ii) specifying the part to which it relates, by giving the part description.

The part description is composed of two labels, since it is useful to be able to separate the part geometry from the part physics, i.e. to be able to specify permittivities and conductivities etc. independently of the shape of the part. The part physics is specified either by giving a class and associated parameters, or as an entity on disk. The part geometry may be specified similarly.

Having specified the parts, or more precisely the multiblocks, it is necessary to describe how they are put together. If two blocks A and B have a common face, then it is only necessary to give the information that a particular face of block A maps to a face in block B. The convention for labelling faces is indicated in Fig 1 which shows how the labels N, S, E, W, U and D (respectively North, South, East, West, Up and Down) relate to the coordinates used within a block

If blocks only meet over part of a face then it is necessary to specify which fraction of each block's face is involved. Incorporating ideas from Ref [2], connectivity is specified via surface patches. Further, it is helpful to consider the

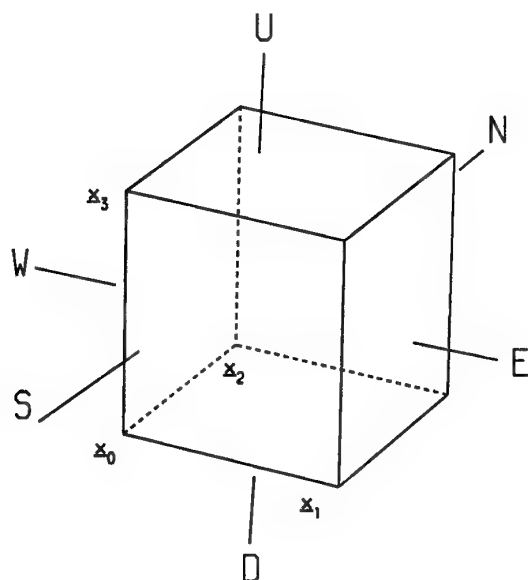


Figure 1: Illustrates the labelling of faces of a multiblock. Position x_0 is the origin of the block co-ordinates and vectors x_1 , x_2 and x_3 indicate the 1, 2 and 3 co-ordinate directions respectively.

co-ordinate system in which the block maps to the unit cube. (Such a system is obtained by a straightforward contraction from the system proposed in Ref [2]). The unit cube system is then used to measure the extent of the surface patch. The surface patch is specified by giving the face label and co-ordinates of a pair of opposite corners (the 'O' and 'X' points). Note the implication that the patch shape in physical space is effectively determined by the surface in which it lies.

To specify connectivity in the input file, a list rather than a NAMELIST is appropriate, since we adopt the convention that for each block, connectivity information for every face must appear. Although this implies a factor of two redundancy in the input, since clearly the same patch information appears in the two entries for the adjacent blocks, validation of the input is considerably simplified.

It is desirable also to specify the order in which the parts are assembled. The main reason is the need to determine how many elements are used in each surface patch. It is easy to see that conflicts can occur. For example in Block A there may have to be say 12×12 element faces in the face common to Block B, because Block A has been gridded using a foreign mesh generator, but the chosen gridding for Block B might be $16 \times 16 \times 16$ elements. The order of assembly can be used to resolve conflicts by preferring data associated with the block that appears earlier in the ranking. In the example shown, if A appears before B, then the code checks to see whether a $12 \times 12 \times 12$ gridding is possible for Block B, which will usually be the case if B is specified as a subprogram class. If B appears before A then the conflict cannot be resolved and an error

message should result.

The above example serves to introduce the concept of elastic grid sizes $n_1 \times n_2 \times n_3$ where n_i is the number of elements in a coordinate direction i . For most blocks, the associated n_i are parameters that can be changed to try to resolve conflicts. The convention adopted is that instead of using positive integers for the n_i in these blocks, a negative integer is used for each co-ordinate direction in which the grid size is effectively arbitrary. It is worth noting that even when all blocks have been collected some n_i may remain elastic, and means are provided for defining these. The default is to replace negative n_i by their absolute values, so that in effect negative n_i suggest grid sizes but do not prescribe them.

3 BASIC INPUT FORMATS

The Section begins by describing the standard tagged input format then goes on to outline other, simpler tagged formats. Detailed specifications for each tag are given in Section 4.

3.1 Standard Format

The standard format of a tagged record as used in Section 4 is written:

```
QQ_label_[howstored=]class_[par1_par2...].
```

Labels are case sensitive and should not contain spaces, commas or colons. If the classname contains spaces, these may be replaced by underscore or the entire name should be enclosed in quotes.

To explain the above, note first the convention that items appearing within square brackets are optional. The three compulsory items, each separated by at least one space “_”, are QQ which is the generic form for a tag, a label which does not occur in any other record with the same tag, and a classname. An example of such a simple record is:

```
BG_part1g_regular_cubic_lattice
```

This creates a block geometry entry that is referenced by the label *part1g* and described as belonging to the class *regular_cubic_lattice* so that e.g. there exists a subroutine CUBREG to generate the particular geometry.

Parameters that determine *part1g* within the class *regular_cubic_lattice* are specified in subsequent lines of input, using NAMELIST as defined in the standard[1]. Thus the complete entry for *part1g* also includes, e.g.

```
&CUBREG
RXMAX = 0.1, RYMAX = 2.0, RZMAX = 3.0/
```

that specifies the dimensions of the cuboid region (in metres).

The optional entries *par1*, *par2*, ... are special parameters that can be set without recourse to the verbosity of NAMELIST. Their meaning depends on the tag. For BG they are the n_i , in order $i = 1, 2, 3$, defining the grid size. Thus a more complicated entry might be

```
BG_part1_regular_cubic_lattice_20_60_40
```


The use of minus denotes the size is elastic (see Section 2), but that a suggested mesh-size for the part is $20 \times 60 \times 40$ elements.

As its name suggests the optional *howstored* entry indicates how the information is stored. If it is omitted a namelist normally follows as in the example. However other values are possible, depending on the class and tag. All tags allow *howstored* to take the value *file*, indicating the file with the name *class* is to be searched for the remaining information associated with the entry. The file *class* will itself contain an entry of the above form.

Values for *howstored* that may be allowed depending on the tag are *number* and *metric*. *Number* implies that the block geometry is specified as a set of numbers representing e.g. the co-ordinates of mesh-points. *Metric* indicates that the metric information needed for the timestepping calculation is already available. The above serve directly or indirectly to define new classes of geometry.

3.2 List of Labels

There are basically two simpler formats for input. One consists of a tag followed by a list of labels, constituting a complete entry. This is used e.g. to define blocks in terms of predefined block geometries and physics thus

```
BL_b11_part1g_part1p
```

and

```
BL_b12_sameas_b11.
```

Note that *sameas* has a special meaning and should not therefore be used as a block label. Any blocks defined without use of a *sameas* can also be thought of as "block types". The same format is used to specify the order of assembly of blocks.

3.3 Connectivity Format

The second simpler input format is used to define connectivities. The tagged line contains only a label (in addition to the tag). Subsequent lines contain the block connectivity information as series of entries (one to a line) in the format:

$$face1[(x_1^{1O}, x_2^{1O})(x_1^{1X}, x_2^{1X})] = block : face2[(x_1^{2O}, x_2^{2O})(x_1^{2X}, x_2^{2X})]$$

terminated by a line containing only the tag EN. Omitting the quantities in square brackets is equivalent to taking

$$(x_1^{iO}, x_2^{iO})(x_1^{iX}, x_2^{iX}) = (0, 0)(1, 1), i = 1, 2.$$

The x^{iO} and x^{iX} are the co-ordinates of the 'O' and the 'X' points defining a patch on the face *face1* where $i = 1$ refers to the block of which the label appears on the tagged line and $i = 2$ refers to other blocks, the labels of which appear immediately to the right of the equals sign.

4 DETAILED INPUT SPECIFICATION

The allowed tags are assigned one of three input formats. The tags BG, BP and PP are used to define respectively the geometry of a block, the physics of a block, and physical data associated with a surface patch (including boundary conditions). The tag CU is used to define a curve within the global geometry. The tag DE is used to set default values for parameters, and SF is used to set parameters that otherwise have no associated tag. The input associated with all these tags (and FL, PO and SC to be defined below) has the standard format.

The tags BL and PA are used to define respectively a block in terms of its geometrical and physical attributes and a patch in terms of its physical attributes. Tag OR is used to specify the order in which blocks should be connected. The input associated with these tags has the list of labels format.

The tag BC is used to define block connectivity information. The information in Section 3.3 constitutes a detailed input specification for this tag, thus no further description appears here.

The tag PS is used to define a plot set, i.e. data that is to be output by the PIC3D for subsequent analysis and plotting. A plot set is defined in terms of labels that identify the fields to be output, the locations at which their value is to be output, and the corresponding times. The first two are defined by list of labels tagged with FS for fields and DS for the set of domains, and the last by tag TS which has the standard format. The plot output of fields may be controlled more tightly by using the FL tag and its associated namelist. The domain set is a list, either of domains specified by the list of labels tagged by DO, or of subdomains. Each domain is made from subdomains that are defined by tag SU that also has the standard format. The importance of the domain concept is that fields are summed over its constituents.

The Table provides a summary of formats for the different tags.

4.1 Tags in Standard Format

4.1.1 BG, BP, CU and PP

Input associated with the tag BG has been specified in Section 3.1. Much the same detailed specification applies to tag BP. The only major differences is that the metric information now relates to permittivity, permeability and conductivity, rather than to position. As might be expected, the associated namelist names also differ. Tag PP input is very similar, with three exceptions, namely (i) that the special parameters are now only n_1 and n_2 since surfaces are two-dimensional, (ii) that it is inappropriate to have direct input of metric terms as a *howstored* option, and (iii) the namelist names are different. Tag CU enables a curve, possibly closed, to be defined by inheritance from a preexisting geometry and usually by extension of a line specified in one block.

4.1.2 DE, FL, PO, SC and SF

All these tags have no special parameters and allow *howstored* only to take the value *file*. If *howstored* is not set, then default values are set for variables in the

Table 1: Tag Table

Tag	Brief Description	Format+	Special Parameter	Notes*
BC	Block Connectivity	C	-	-
BG	Block Geometry	S	N_1, N_2, N_3	f n m
BL	Block definition	L	-	3 labels
BP	Block Physics	S	N_1, N_2, N_3	f n m
CU	CURve definition	L	-	-
DE	DEfaults (namelists)	S	-	f
DO	DOmain	L	-	any number of labels
DS	Domain Set	L	-	any number of labels
FL	Field data	S	-	f
FS	Field Set	L	-	any number of labels
OR	ORdering of blocks	L	-	any number of labels
PA	PAch definition	L	-	3 labels
PO	Parameter StOrage	S	-	f n
PP	Patch Physics	S	N_1, N_2, N_3	f n
PS	Plot Set	L	-	4 labels
SC	Signal Component	S	-	f
SD	Secondary Domains	L	-	any number of labels
SF	Set Free parameters	S	-	f
SG	SiGnal	L	-	-
SP	SPecies definition	S	$N_1 - N_5/R_1 - R_5$	f n
SU	SUBdomain	S	$N_1 - N_3, R_1 - R_6$	f
TS	Time Set	S	$N_1 - N_6/R_1 - R_6$	f

*f, n and m are abbreviations for the allowable *howstored* options *file*, *numbers* and *metric*.

+S, L and C denote the standard, the list of labels and the connectivity formats respectively

namelist that follows in the case of DE.

For PO, parameters are set defining the initial magnetic field, initial current, source current, particle beam, initial electric field, external magnetic field or external electric field, depending whether the label is set to **bfield**, **current**, **jsource**, **pbeam**, **efield**, **bext** or **eext** respectively. The parameters defined are a subset of NAMELIST *SF*, namely

AMODE(3)	box size for modes
BUNI(3)	uniform initial B
CDEN0	beam number density
CMODE(2)	mode amplitudes
CP0	beam momentum in z
FMODE	frequency of mode
NINIT	initial field type
NMODE(3)	mode numbers
RPHSHF(3)	phase shifts (in radians)
ZMODE	impedance of mode

Negative NINIT set an initial current that lasts for the first timestep only (a "ping"), the spatial dependence being determined by the absolute value as follows:

1. an initial uniform magnetic field
2. an initial cylindrical TE mode
3. a cylindrical TM mode
4. a rectangular TE mode
5. a rectangular TM mode
6. a magnetostatic (PPM) mode.
7. a spiky mode, a sum over four axial harmonics of a TM mode
8. a helical ping of z component
9. a cylindrical ping of z component

The field components are set by BUNI and mode parameters are set by variables ending in MODE. Clearly only simple device geometries are allowed, and the 3-coordinate must correspond to z.

For FL, NAMELIST *FL* allows the setting of the variable LFSKIP to control particle skip length for the field given by CFLNAM and other parameters as listed

CFLNAM*64	diagnostic field name
FMAX	diagnostic field maximum
FMIN	diagnostic field minimum

FUNIT	diagnostic field unit
FXMAX	diagnostic x-field maximum
FXMIN	diagnostic x-field minimum
FXUNIT	diagnostic x-field unit
FXZERO	diagnostic x-field zero
FZERO	diagnostic field zero
LFCOLO	diagnostic field colour
LFSKIP	diagnostic field particle skip

For SC a component of a signal is specified via NAMELIST *SIGNAL* which contains the variables

FREQUENCY	signal component frequency in Hertz
DURATION	component duration in s
SIGNAL_FORM*8	identifies form of signal component
PHASE	phase of signal component (degrees)

The allowed SIGNAL_FORMs are 'CONSTANT', 'LINEAR', 'RISE', 'PLATEAU' and 'S-SHAPE'. For SF, depending on *class*, parameters are set that otherwise have no associated tags.

4.1.3 TS and SU

The time set specifies the times at which output is required:

TS timeset-name class [start-time end-time output-interval average-time time-zero time-unit].

The *time-unit* is the negative of the base ten exponent of the scale factor for the times, i.e. 9 implies time measured in nanoseconds. If the *time-unit* is zero and all other times are integral, then the unit of time is taken to be the timestep. If no times are set then output will take place at every timestep to a maximum set by NRUN. The value of *class* is arbitrary.

The SUBdomain specifies a cuboidal subset of a block

SU subdomain-name block-name [d o x]

The *block-name* is the name of a block, *o* and *x* are the positions of the corners of the subdomain in coordinates on $[0, 1]^3$. If *o* and *x* are omitted the entire block is assumed. *d* is a triple of the integers (1, 2, 3) which gives directions for evaluating components and integrals. *d*(1) gives the direction for line integrals, ie the integral is taken in the direction $\bar{x}_{d(1)}$, and if the subdomain extends normal to *d*(1), a set of line integrals one for each element in the normal plane will be computed. For surface integrals, the area element is proportional to $d\bar{x}_{d(1)}d\bar{x}_{d(2)}$ and a set may also be computed. If *d* is omitted then *d*(i) = i, i = 1, 2, 3. Experts may take the *d*(i) negative and set *d*(3). There are no namelists corresponding to either TS or SU.

4.1.4 SP

Tag `SP` defines a species

```
SP species-name class mass charge number-
particles exponent storage time-step
```

where the species `mass` is in units of the electron mass, `charge` is units of the absolute charge of an electron and `number-particles` is the number of particles on a species superparticle, the `exponent` of which is given separately. The variable `storage`, if not greater than 100, is the number of particles per element, and `time-step` is in units of the electromagnetic timestep. For example, the electron species line might read

```
SP electron any 1 -1 1 9 8 2
```

if there are 10^9 charges on a superparticle, 8 particles are expected per element and the subcycling parameter `NSUBS` = 2.

Further particle data may be specified using the NAMELIST `PCLE`, for the classes `below` and `extra`, see Section 5.3.

4.2 List of Labels Tags

4.2.1 BL and PA

Tags `BL` and `PA` define a multiblock and a surface patch respectively. The format for `BL` is specified in Section 3.2, and that for `PA` differs only in that the label following the patch identifier is usually *sameas*, thus an example of a record is:

```
PA pa1 sameas perfect_conductor1
```

where the patch physics associated with label *perfect_conductor1* has already to have been defined using an item tagged with `PP`. However, if boundary conditions involving particles are to be set, the second label should point to an item tagged with `PP` that defined the patch particle physics.

4.2.2 OR

The tag `OR` is followed by a list of block labels in the order of assembly, so that e.g.

```
OR blz blc bla
```

indicates that information associated with block *blz* supersedes information associated with block *blc* which in turn supersedes information about block *bla*. Equivalently think of putting *blz* on the workbench, then attaching *blc* and then *bla*.

4.2.3 PS

A Plot Set is specified by a `PS` tag followed by a file format name - `tsd` for time series data, `gr1` for line plots, `gst` for particle scatter plots, and `pnt`, `lin` and `pts` to save time series, line plot and scatter plot data respectively for

subsequent processing. There then follows the name of a Field Set, a Domain Set and a Time Set. The format is

PS_{file-format}_{Field-Set}_{Domain-Set}_{Time-Set}.

4.2.4 FS

The Field Set specifies a list of fields

FS_{fieldset-name}_{field-name1}_{field-name2}_{field-name3}.....

The names of fields that are allowed are:

1. E1, E2, E3, H1, H2 and H3 representing the components of the electromagnetic fields \mathbf{E} and \mathbf{H} in the local block co-ordinate system.
2. intE.dl and intH.dl representing the line integrals $\int \mathbf{E} \cdot d\mathbf{l}$ and $\int \mathbf{H} \cdot d\mathbf{l}$ respectively. Note there are no spaces following the string int.
3. intD.dS, intB.dS and intj.dS representing the surface integrals $\int \mathbf{D} \cdot d\mathbf{S}$, $\int \mathbf{B} \cdot d\mathbf{S}$ and $\int \mathbf{j} \cdot d\mathbf{S}$ respectively. Note there are no spaces following the string int.
4. intE.D/2dV, intB.H/2dV and intj.EdV representing the volume integrals of $\frac{1}{2}\mathbf{E} \cdot \mathbf{D}$, $\frac{1}{2}\mathbf{B} \cdot \mathbf{H}$ and $\mathbf{J} \cdot \mathbf{E}$ respectively. Note there are no spaces in any of these labels.
5. Y(X) where X and Y are global coordinates for 2-D scatterplots as follows:
 x - x coordinate, y - y coordinate, z - z coordinate, r - cylindrical polar r coordinate, theta - cylindrical polar θ coordinate, p1 - physical component of momentum in the 1 mesh direction, p2 - physical component of momentum in the 2 mesh direction, and p3 - physical component of momentum in the 3 mesh direction.
6. a label appearing in a line containing the FL tag.

4.2.5 DO

The DObain specifies a list of subdomains

DO_{domain-name}_{subdomain-name1}_{subdomain-name2}.....

Note that fields are accumulated over elements of a domain. Primary subdomain order is important, and in particular the first sub-domain must be a primary.

4.2.6 DS

The Domain Set specifies a list of domains or subdomains

DS_{domain-set-name}_{domain1}_{subdomain1}.....

4.2.7 SD

The tag SD defines the subdomains which are secondary to the first listed subdomain within the domain `domain-name`.

`SD_domain-name_primary-subdomain_secondary-subdomain_.....`

A maximum of 30 secondary subdomains are allowed

4.2.8 SG

SG specifies a signal as a list of components

`SG_signal-name_cpt1_cpt2_....`

Recognised names of signals are `boundary` for setting the time varying amplitude of electromagnetic boundary conditions, `current` and `rfcurrent` similarly for electric current forcing, and `inject` similarly for injected particle beams.

5 CLASS DESCRIPTIONS

5.1 Block Geometry

5.1.1 Rectbk

This is the name of the base subroutine used to mesh orthogonal geometries, that have either a Cartesian reference coordinate system (`MCTYPE = 1`), which is the default case, or a cylindrical polar reference coordinate system (`MCTYPE = 2`). Twelve edge curves defined by discrete points along their length are required to define the block. `MTYPE` is zero for uniform spacing of the grid in curvilinear coordinates. The variables in NAMELIST *RECTBK* are

<code>EDGES(MEDGE)</code>	array of edge curves
<code>MCTYPE</code>	reference coordinate type
<code>MEDGE</code>	dimension of <code>EDGES</code> array
<code>MTYPE(3)</code>	input meshtype selector

5.1.2 Genblk

This is the name of the base subroutine used to mesh nonorthogonal geometries, that have either a Cartesian reference coordinate system (`MCTYPE = 1`), which is the default case, or a cylindrical polar reference coordinate system (`MCTYPE = 2`). Twelve edge curves defined by discrete points along their length are required to define the block. `MTYPE` is zero for uniform spacing of the grid in curvilinear coordinates. The variables in NAMELIST *GENBLK* are

<code>EDGES(MEDGE)</code>	array of edge curves
<code>MCTYPE</code>	reference coordinate type
<code>MEDGE</code>	dimension of <code>EDGES</code> array
<code>MTYPE(3)</code>	input meshtype selector

5.1.3 Parallelepiped

A parallelepiped is specified by the lengths of its three sides and the (spherical) polar angles of two of them, the third side being assumed to lie along the polar axis z . The variables in NAMELIST *PIPED* are

RPHI1	azimuthal angle of parallelepiped side 1 (degrees)
RPHI2	azimuthal angle of parallelepiped side 2 (degrees)
RSID1	length of parallelepiped side 1
RSID2	length of parallelepiped side 2
RSID3	length of parallelepiped side 3
THET1	polar angle of parallelepiped side 1 (degrees)
THET2	polar angle of parallelepiped side 2 (degrees)

5.1.4 Polar with regular meshing

This geometry uses a cylindrical polar reference coordinate system. The variables in NAMELIST *POLREG* are

AXMAX	maximum along cyl polar-axis
RADINR	inner radius
RADOUT	outer radius
THEMAX	maximum theta (degrees)

5.1.5 Variable spaced polar lattice

This geometry uses a cylindrical polar reference coordinate system. The variables in NAMELIST *POLVAR* are

AXMAX	maximum along cyl polar-axis
RADINR	inner radius
RADOUT	outer radius
THEMAX	maximum theta (degrees)

5.1.6 Variable spaced cubic lattice

The variables in NAMELIST *CUBVAR* are

RXMAX	maximum along x-axis
RYMAX	maximum along y-axis
RZMAX	maximum along z-axis

5.1.7 Polar mesh segment

The variables in NAMELIST *POLSEG* are

AXMIN	minimum along cyl polar-axis
AXMAX	maximum along cyl polar-axis
RADINR	inner radius
RADOUT	outer radius
THEMIN	minimum theta (degrees)
THEMAX	maximum theta (degrees)

5.1.8 Polar to rectangular transition

The variables in NAMELIST *POLRCT* are

AXMIN	minimum along cyl polar-axis
AXMAX	maximum along cyl polar-axis
RADCUR	radius of curved edge
RADSTR	radius of straight edge
THEMIN	minimum theta (degrees)
THEMAX	maximum theta (degrees)

5.1.9 Regular cubic lattice

The variables in NAMELIST *CUBREG* are

RXMAX	maximum along x-axis
RYMAX	maximum along y-axis
RZMAX	maximum along z-axis

5.1.10 Quadrilateral cylinder

This geometry is the extrusion of a quadrilateral by a length RZMAX. In a plane $z = \text{constant}$, the corners of the quadrilateral are in clockwise order: (0,0), (x1,y1), (x2,y2) and (RXMAX,RYMAX). The variables in NAMELIST *QUADRI* are

RXMAX	maximum along x-axis
RYMAX	maximum along y-axis
RZMAX	maximum along z-axis
XQUAD1	x1 point for quadrilateral cylinder
XQUAD2	x2 point for quadrilateral cylinder
YQUAD1	y1 point for quadrilateral cylinder
YQUAD2	y2 point for quadrilateral cylinder

5.2 Block Physics

5.2.1 Uniform

The variables in NAMELIST *UNIFRM* are

EPSR	relative permittivity in uniform block
RMUR	relative permeability in uniform block

5.3 Particle Properties

5.3.1 Extra and Below

These classes share the same NAMELIST. Class **extra** is provided so that just two parameters, controlling the particle pushing algorithm may be set. The first parameter **NPBGEO**, when set to unity, forces the particle patch buffers to have the format which is consistent with the particle pushing in general geometries which have an underlying cartesian coordinate system, otherwise it is assumed that all blocks have an underlying cylindrical coordinate system. The second of these, **MALG** has a more general role, but the array entries relevant to particle motion are as follows.

- 2** number of corrector steps, except in cylindrical geometry (see **MALG**(18) below).
- 3** number of backward steps taken to try to get the “stagger” right on the first point of a particle orbit.
- 16** if set to unity, use a stepwise approximation to h_2 in cylindrical geometry.
- 18** if greater than unity, sets the number of corrector steps in cylindrical geometry.
- 19** if set to unity, do not update the electromagnetic field (useful for orbit calculations).
- 20** if set to unity, suppress header output in **gst** and **pts** plot sets (useful if such information is to be combined to represent an orbit).

When the class label is set to **below**, a simulation will **BEGIN** with a **LOW** number of particles in one named block of the domain, typically for orbit tests or to stimulate multipaction effects. Positive **NRANDP** is the number of particles introduced with uniformly randomly distributed velocities subject to a maximum speed **SPEEDM** given in units of $c = 3 \times 10^8 \text{ m/s}$. If **NRANDP** is zero, the user may specify each particle separately. The number of particles is then set implicitly – a negative first positional coordinate **UCPOS** implies the end of the particle list. The variables in NAMELIST **PCLE** are

CBLOCK*32	block label
MALG(20)	algorithm parameters
NCVEL	coordinate system for pcle vel
NPBGEO	particle buffer geometry
NRANDP	number of particles in random start
NODIM	number of position coordinates
SPEEDM	max speed of random start particles

UCPOS(3,100) particle position (unit cube)
 VEL(3,100) particle velocity (m/s global)

5.4 Patch Physics

5.4.1 Perfect conductor, Polar axis and Reflector

These classes have no need for an associated namelist.

5.4.2 Applied field, Resistive wall and Surface field

These classes share the same NAMELIST. In the case of an applied electric displacement, the components of field are relative to the coordinates of the block; note that only in some circumstances is \mathbf{d} constant. STHETA corresponds to θ in Ref [7]. Surface field implies a boundary condition of resistive wall plus applied field. The variables in NAMELIST *BCS* are

DAPLYA(3)	applied electric field marker
DPOT	potential difference in V across block (DAPLYA marks direction)
EAMP	electric field amplitude in Vm^{-1}
NGLOB	if 1 use ic as bc, if 0 use applied field
SFORM*8	description of signal
SPOWER	signal power
STHETA	lagging factor for resistive wall bc
SURFZ	surface impedance

DAPLYA sets the phase shifts RPHSHF in the ic used as bc case. It should be a vector normal to the boundary so that a null of electric field \mathbf{d} is avoided.

5.4.3 Emitter, Inject beam and Multipaction

These classes all use the same NAMELIST to specify particle emission on a patch. The relevant variables for space charge limited emission are ALFA, CSPECI, EFFE and NEMALG, whilst those characterising beamed emission are CSPECI, CURBEA, RADBEA, ROTBEA and VBEAM. The multipaction or secondary emission parameters are defined in Ref [8]. The variables in NAMELIST *PARBCS* are

ALFA	emission splitting parameter
CSPECI*32	species name
CURBEA	beam current
DELTAM	multipaction parameter δ_{max}
EFFE	effective E factor
EMCRIT	critical energy for multipaction emission
EME1	multipaction parameter E_1
EME2	multipaction parameter E_2
EMMAX	multipaction parameter E_{max}

EMMEAN	mean energy of inelastic emission
EMSTDD	std dev. energy of inelastic emission
NEMALG	emission algorithm (2 for 2-D)
NERAND	random emission and/or specular reflection
NHYPER	multipaction splitting parameter
NLSEDA	true for use of data array
NLSEE	true for variable emission coefficient
RADBEA(2)	beam radii (inner then outer)
ROTBEA	beam rotation
SIGMES(200)	multipaction parameter array
VBEAM	beam voltage

5.5 Other Geometry

5.5.1 Inherit

This defines a geometrical object by inheritance, ie as parts of predefined objects. Normally the object is also defined by extension, so that an object that crosses many block boundaries can be defined from say a line in one given block. The relevant variables are set using NAMELIST *INHERI* which contains

CBLOCK*32	block label
CUNIVS(512)*32	block labels of universe
NBRANC(2)	branching directions (defines paths)
NDIR(3)	direction markers
NLBOTH(3)	true for both ways in direction
NLOXS	true if o and x points in universe
NLXTEN	true to extend geometry
OPOINT(3)	o-point real coords
OUNIVS(3,512)	o-points in universe
XPOINT(3)	x-point real coords
XUNIVS(3,512)	x-points in universe

5.6 Miscellaneous

5.6.1 Free parameters

This provides the opportunity to designate a specific block to provide the origin of coordinates (using those variables ending in GLB), to set parameters to control a run (NRUN, MDPART, NOPSEL and NXPTDD), and to control diagnostic output as described in Ref [6]. MDPART = 1 prevents the introduction of particles, but both it or NBEXT will be overridden by any use of respectively the SP or PO tags. The value of NXPTDD has the following effect:

1. print EHWSP before and after patch exchange
2. print D and EHWSP before and after 2nd patch exchange

3. particle diagnostics including positions
4. print B, D and EHWSP
5. use with debugger
6. dump EHWSP on every call of subroutine EXPT
7. print B and D
8. dump C on every call of subroutine EXPT
9. print total number of particles in each block every step

DTMUL is the Courant number, which multiplies the right-hand side of equation (38) of Ref [9].

The variables in NAMELIST *SF* which have not already been described in Ref [6] or in the section for tag PO, are

CBLGLB*32	block with designated origin and rotation
DTMUL	timestep multiplier
MALG	if entry 17 is unity, discrete correction to ic
MDPART	Dimension of PARTicle coordinate arrays
NBEXT	use initial B as external B
NLDUMP	true for binary dump
NOPSEL	Select Output Sequence
NPAR(MAXPAR)	integer parameter defaults on tagged line
NRUN	number of time steps
NTPRES	number of processes needed on target
RFIGLB	spherical phi rotn of named block
RPAR(MAXPAR)	real parameter defaults on tagged line
THEGLB	spherical theta rotn of named block
XYZGLB(3)	physical position of named block

References

- [1] Document ISO/IEC 1539:1991E.
- [2] J W Eastwood, R W Hockney and W Arter, *General Geometry PIC for MIMD Computers: Final report*, Report RFFX(93)56, Culham Laboratory, June 1993.
- [3] W Arter, *The 3-D General Geometry PIC Software for Distributed Memory MIMD computers: Description of Inputs used to Generate Meshes*, AEA/TLNA/31876/TN/4, January 1994.
- [4] W Arter, *The 3-D General Geometry PIC Software for Distributed Memory MIMD computers: Extended Description of Inputs used to Generate Meshes and Control Runs*, AEA/TLNA/31878/TN/14, October 1994.

- [5] W Arter, *The 3-D General Geometry PIC Software : The Current State of the PEGGIE Preprocessor*, AEA/TYKB/31777/TN/1, May 1994.
- [6] N J Brealey, J W Eastwood and W Arter, *3DPIC Diagnostics*, Culham Laboratory Technical Note AEA/TYKB/31876/TN/9, September 1994.
- [7] W Arter, *Boundary Conditions for Maxwell's Equations in General Geometry*, AEA/TYKB/31876/TN/11, September 1994.
- [8] W Arter, *Three-dimensional Multipaction in General Geometry*, Culham Laboratory Technical Note AEA/TYKB/28006/TN/3, September 1995.
- [9] W Arter, *Dispersion and Stability analysis for Maxwell's Equations*, AEA/TYKB/31876/TN/8, September 1994.

E AEA/TYKB/28006/TN/5

AEA/TYKB/280000/TN/5

MPICTIM RELEASE 2.10.00:
USER GUIDE

N J Brealey

September 1995

AEA Technology
Culham Laboratory
Abingdon
Oxfordshire
OX14 3DB

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data clause at DFARS 252227-7013.

AEA Technology, Culham Laboratory, Abingdon, Oxfordshire, OX14 3DB, England.

Document Control Number: AEA/TYKB/280000/TN/5			
Date of Issue: September 1995			Issue number: 1
<i>Authorization</i>	<i>Name</i>	<i>Signature</i>	<i>Position</i>
Prepared by	N J Brealey	<i>N.J. Brealey</i>	Project Scientist
Checked by	W Arter	<i>W. Arter</i>	Project Scientist
Approved by	J W Eastwood	<i>J.W. Eastwood</i>	Project Manager

Contents

1	New Features in MPICTIM Version 2.10.00	3
2	Using MPICTIM	3
2.1	Starting MPICTIM	3
2.2	Quitting MPICTIM	5
2.3	Opening Data Files	5
2.4	Setting Processing Options	5
2.5	Choosing the Data to be Processed	6
2.6	Producing Plots	7
2.7	Annotating Plots	7
2.8	Saving and Erasing Plots	7
2.9	Viewing Parameters	8
2.10	Extracting Data from the Plot	8
3	Components of MPICTIM	9
3.1	Menu Bar	10
3.1.1	File Menu	10
3.1.2	View Menu	10
3.2	Plot Controls	10
3.2.1	Plot Button	10
3.2.2	Label Field	11
3.2.3	Colour Option Menu	11
3.2.4	Annotate Button	11
3.2.5	Erase Button	11
3.2.6	Save Button	11
3.2.7	File Name Field	11
3.3	Processing Controls	11
3.3.1	Period Field	11
3.3.2	Diff Toggle Button	12
3.3.3	Processing Option Menu	12
3.3.4	Int Toggle Button	12
3.3.5	Graticule Toggle Button	12
3.3.6	Log Toggle Button	12
3.4	Audit Data Selection Area	12
3.5	Surface Data Selection Area	13
3.6	Ranges Area	13
3.6.1	Time Range	13
3.6.2	Frequency Range	13
3.6.3	Variable Range	13
3.7	Labels Region	14
3.8	Buttons Area	14
4	Installation Notes	14
4.1	Solaris 2.x Distribution	14

MPICTIM RELEASE 2.10.00: USER GUIDE

N J Brealey

September 1995

Abstract

The MPICTIM code is designed to examine time series data produced by PIC simulation codes. It uses an OSF/Motif [1] Graphical User Interface (GUI) instead of the character based interface; this OSF/Motif interface to MPICTIM is much easier than a Command Line Interface (CLI) for interactive use.

1 New Features in MPICTIM Version 2.10.00

This User's Guide describes MPICTIM Version 2.10.00 which has replaced MPICTIM Version 1.00.00 which was described in [2].

The following new features have been added to MPICTIM:

1. Values can be extracted from the plot by using the Get Values button and clicking on the plot with the left mouse button.
2. Plots using logarithmic scales can be produced by activating the Log toggle button.
3. Graticules can be produced by activating the Graticule toggle button.

2 Using MPICTIM

This Section contains a task orientated description of how to use MPICTIM. Section 3 describes the functions of the different components of MPICTIM. The MPICTIM application uses OSF/Motif Graphical User Interface components. You should refer to your system's documentation for general information on using OSF/Motif. This document concentrates on aspects which are specific to MPICTIM.

2.1 Starting MPICTIM

To start MPICTIM start an `xterm` window and issue the `mpictim` command at the prompt:

```
$ mpictim
```

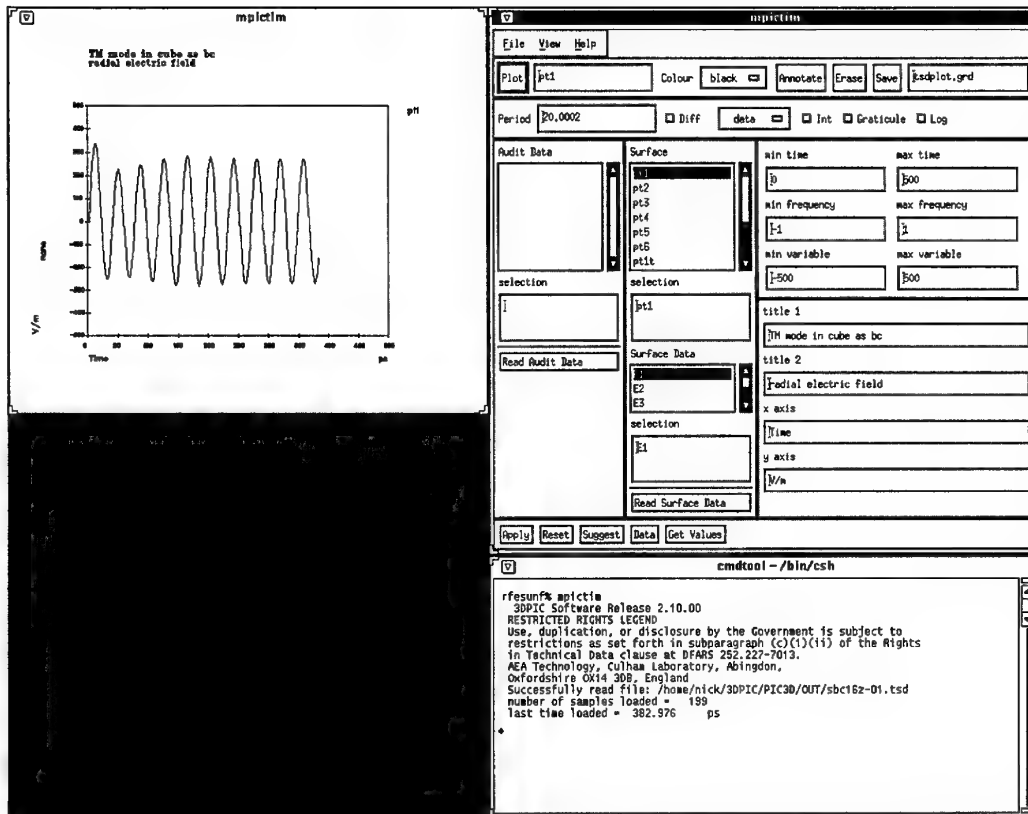


Figure 1: Motif Time Series Analysis Program MPICTIM

Move and resize the `xterm` window to the lower right of the screen as shown in Figure 1. Your workstation screen should now look like the screen shown in Figure 1. You should consult your system administrator if it looks substantially different.

2.2 Quitting MPICTIM

To quit the application activate the File pull down menu from the menu bar (Figure 2) and select the Exit option.

2.3 Opening Data Files

To open a time series data file activate the File pull down menu from the menu bar (Figure 2) and select the Open option to pop up the Open File dialogue window (Figure 3). Use the Open File dialogue window load a file by selecting a file and activating the OK button. Opening a time series data file replaces all time series data held in the code. If you have sequential data from restart calculations which occupies several files use should use the Append option from the File menu to append data without deleting data which is already loaded.

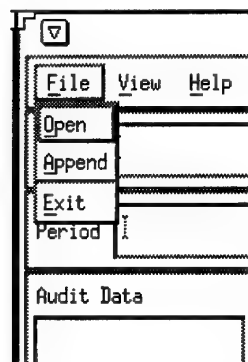


Figure 2: File Menu

2.4 Setting Processing Options

The data processing options are set in the Processing Options area. The options take effect when a data item is selected and either of the Read Audit Data or Read Surface Data buttons are activated.

The period used by the mean, rms, variance and spectrum plots is set by typing text in the Period text field and activating the Apply button at the bottom of the main window.

The Diff toggle button enables differentiation of the data before other processing is performed.

The Processing option menu (Figure 4) allows the user to choose how the data is processed.

The Int toggle button enables integration of the data after other processing has been performed.

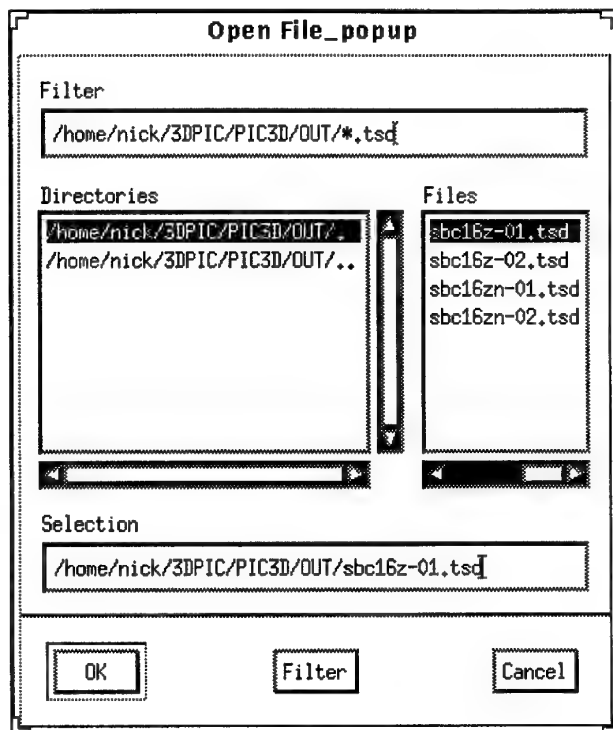


Figure 3: File Selection Dialogue

The Log toggle button enables logarithmic scale plots.

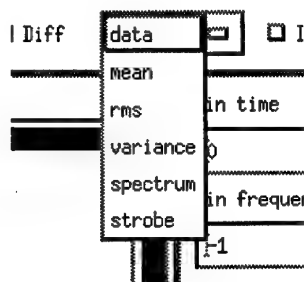


Figure 4: Data Menu

2.5 Choosing the Data to be Processed

To choose the data to be processed select an audit variable from the Audit Data scrolling list or select select a surface and a surface variable from the Surfaces and Surface Data scrolling lists.

Activate the Read Audit Data button or the Read Surface Data button to process the data.

2.6 Producing Plots

The user may select the colour used for plots using the Colour option menu (Figure 5). The plot label is set in the Label next to the Plot button. The user should set the ranges for the plots in the ranges fields. The Apply button at the bottom of the main window must be activated to make the values active. The time range is also used as a window when processing data. The Reset, Suggest and Data buttons at the bottom of the main window may be used to display current ranges, suggested ranges and data ranges (and other parameters).

Once the parameters are set activating the Plot button displays the plot on the screen.

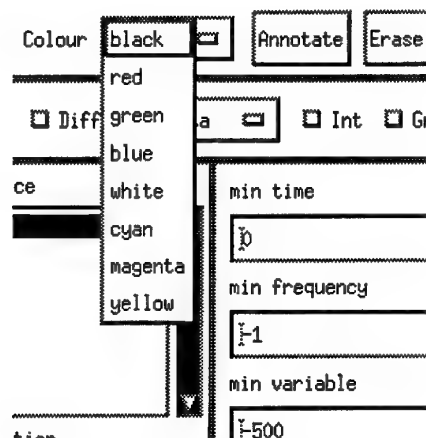


Figure 5: Colour Menu

2.7 Annotating Plots

The labels used for annotating the plot are set in the labels text fields. The Apply button at the bottom of the main window must be activated to make the labels active. The Reset, Suggest and Data buttons at the bottom of the main window may be used to display current labels, suggested labels and labels from the data file (and other parameters). The Graticule toggle button enables the plotting of graticules.

Once the labels are set activating the Annotate button annotates the plot on the screen and plots the axes and graticule (if enabled).

2.8 Saving and Erasing Plots

To save a plot activate the Save button. The plot is saved to a GHOST GRID file [3] with the name given in the text field immediately to the right of the save button.

To erase a plot activate the Erase button.

2.9 Viewing Parameters

Information from the time series data files can be displayed in the window from which MPICTIM was started by activating one of the options from the View pull down menu on the menu bar (Figure 6).

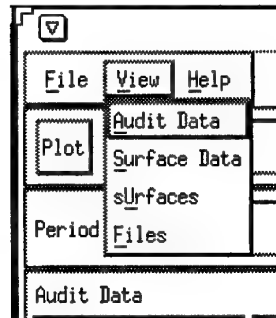


Figure 6: View Menu

2.10 Extracting Data from the Plot

To extract data from the plot activate the Get Values button at the bottom of the main window. The controls will then be grayed out to indicate that they are inactive and a message similar to the following will be printed in the `xterm` window:

```
Collecting values from graph:
click mouse left button for data, right for done
```

```
x: Time          ps
y: E1           none
```

The text after the `x:` and `y:` gives the name and units of the current variables being plotted.

Each time the left mouse button is clicked at a point on the plot the coordinates are printed in the `xterm` window using the current scales: E.g.

```
x = 192.71    y = 9.9188
x = 249.40    y = -1.9838
x = 370.25    y = -77.570
```

When the right mouse button is clicked on the plot the controls on the main window become activate and are ungrayed and the following message is printed in the `xterm` window

```
Data collection finished
```

3 Components of MPICTIM

This Section describes the functions of the different components of MPICTIM. Section 2 contains a task orientated description of how to use MPICTIM.

Figure 1 shows the two base windows of the MPICTIM application. The main control window is top right and the graphics display window is top left. The terminal window from which MPICTIM was started is bottom right.

The main control area is shown in greater detail in Figure 7. The components of this window are shown in greater detail in Figure 7 and are described in the following subsections. The components are described in order starting from the top left and scanning from left to right and from top to bottom.

The screenshot shows the MPICTIM main control window. It features a menu bar with 'File', 'View', and 'Help'. Below the menu bar is a toolbar with buttons for 'Plot', 'pt1', 'Colour' (set to 'black'), 'Annotate', 'Erase', 'Save', and 'itsdplot.grd'. A row of controls includes 'Period' (20.0002), 'Diff' (checkbox), 'data' (dropdown), 'Int' (checkbox), 'Graticule' (checkbox), and 'Log' (checkbox). The main area is divided into several sections: 'Audit Data' with a list box containing 'pt1', 'pt2', 'pt3', 'pt4', 'pt5', 'pt6', 'pt1t'; 'Surface' with a list box containing 'pt1'; 'min time' (0) and 'max time' (500); 'min frequency' (-1) and 'max frequency' (1); 'min variable' (-500) and 'max variable' (500); 'title 1' (TM mode in cube as bc); 'title 2' (Radial electric field); 'x axis' (Time); and 'y axis' (V/m). At the bottom are buttons: 'Apply', 'Reset', 'Suggest', 'Data', and 'Get Values'.

Figure 7: Main Window

3.1 Menu Bar

The menu bar contains two pull down menus: the File menu and the View menu.

3.1.1 File Menu

The File menu contains three items:

Open pops up a file selection dialogue to load data from a time series data file. Loading the data file replaces any data already loaded.

Append pops up a file selection dialogue to append data from a time series data file to data already loaded. The time series data file must contain the same data items and use the same time sampling interval as the files which are already loaded.

Exit quits the application.

3.1.2 View Menu

The View menu is used for displaying information in the window from which MPICTIM was started. The View menu contains four items:

Audit Data displays a list of audit variables. The list gives the variable numbers, the labels, the units used and the long names.

Surface Data displays a list of surface variables. The list gives the variable numbers, the labels, the units used and the long names.

surfaces displays a list of surfaces. The list gives the variable numbers, the labels, the default plot colours and the long names.

Files displays a list of the files that have been loaded. The list gives the file name, the run identification string, the sample range in steps, the sample range in units of time, the hinted period in steps and the sampling interval.

3.2 Plot Controls

The plot controls region contains controls used to plot processed data. Working from left to right the controls in this area are:

3.2.1 Plot Button

Pressing the plot button plots the processed data on the screen using the applied ranges. The plot is labelled with the text from the Label field (immediately to the left of the plot button) and is plotted in the colour set in the Colour option menu.

3.2.2 Label Field

The label field is immediately to the left of the plot button. The text in this field (up to 8 characters) is used as the label for the current curve being plotted. This field may be made blank if no label is required.

3.2.3 Colour Option Menu

This control sets the colour used to plot a curve. The colour choices are black, red, green, blue, white, cyan, magenta, yellow. (White will not be visible because the background is white).

3.2.4 Annotate Button

This button annotates the plot with the applied labels from the Labels Area. Two title lines and labels for the axes are added.

3.2.5 Erase Button

This button erases the plot.

3.2.6 Save Button

This button saves the plot to a file with the name given in the file name field immediately to the left and erases the plot.

3.2.7 File Name Field

The file name field is immediately to the left of the Save button. It may contain up to 16 characters of text. It specifies the name of the GHOST grid file in which the current plot will be stored. The file is placed in the directory in which MPICTIM was started.

3.3 Processing Controls

This area contains controls which determine how a data item is processed when it is read.

3.3.1 Period Field

This field is used for setting and displaying the period used by the mean, rms, variance and spectrum processing options. The period is set when the Apply button in the Buttons area at the bottom of the application is activated. The hinted period from the data file is shown when the Data button is activated, the suggested period is shown when the Suggest button is activated and the current value is when the Reset button is activated.

3.3.2 Diff Toggle Button

When the Diff Toggle button is active the data will be differentiated with respect to time before other processing takes place.

3.3.3 Processing Option Menu

This option menu selects the type of processing applied to the data when it is processed. There are six options available:

data is unchanged

mean computed over current period

rms (root mean squared) values computed over current period

variance computed of current period

spectrum computed for time interval truncated to an integral multiple of the current period

stroke plot computed for time interval truncated to an integral multiple of the hinted period

3.3.4 Int Toggle Button

When the Int Toggle button is active the data will be integrated with respect to the independent variable after other processing has taken place.

3.3.5 Graticule Toggle Button

When the Graticule Toggle button is active a graticule will be plotted when the Annotate button is activated.

3.3.6 Log Toggle Button

When the Log Toggle button is active the data will be plotted using logarithmic scales when the Plot button is activated.

3.4 Audit Data Selection Area

This area is used to select an Audit Data item and process it using the current options. The user selects the variable in the scrolling list, the long label is then shown in the selection text field and the data is processed by activating the button labelled "Read Audit Data". The processed data remains unchanged until another Audit Data item or a Surface Data item is read.

3.5 Surface Data Selection Area

This area is used to select an Surface Data item and process it using the current options. The user selects the surface and the variable in the scrolling lists, the long labels are then shown in the selection text fields and the data is processed by activating the button labelled "Read Surface Data". The processed data remains unchanged until another a Surface Data item or Audit Data item is read.

3.6 Ranges Area

This area contains text fields for viewing and setting various ranges. The values in the text fields are used to set the ranges when the Apply button in the Buttons area at the bottom of the window is activated. The text fields are set to the current values when the Reset button is activated, they are set to suggested values when the Suggest button is activated and are set to the data ranges (after processing) when the Data button is activated.

3.6.1 Time Range

These text fields may show the time range of the processed data, the suggested time range of the processed data, the current time range or a range which may be applied, depending on which of the Apply, Reset, Suggest or Data buttons have been activated. This interval is used as a data window during the processing of data i.e. data outside the interval is discarded. This interval is also used as the time interval for plots which have time as the independent variable.

3.6.2 Frequency Range

These text fields may show the frequency range of the processed data, the suggested frequency range of the processed data, the current frequency range or a range which may be applied, depending on which of the Apply, Reset, Suggest or Data buttons have been activated. This interval is used as the frequency interval for plots which have frequency as the independent variable. It is also used as the interval for the strobe plots in which data from an interval is processed to produce one period of the signal as a function of time. (For Strobe plots the time axis is in units of the hinted period).

3.6.3 Variable Range

These text fields may show the dependent variable range of the processed data, the suggested dependent variable range of the processed data, the current dependent variable range or a range which may be applied, depending on which of the Apply, Reset, Suggest or Data buttons have been activated. This interval is used as the dependent variable interval for all plots.

3.7 Labels Region

This area contains text fields for viewing and setting various labels used for annotating the plots. The values in the text fields are used to set the labels when the Apply button in the Buttons area at the bottom of the window is activated. The text fields are set to the current values when the Reset button is activated, they are set to suggested values when the Suggest button is activated and values from the data file when the Data button is activated.

Two lines of titles and labels for the x and y axes may be set and viewed.

3.8 Buttons Area

This area contains four buttons which control parameters used by the application:

Apply This button takes the parameters from the Ranges area, the Labels area, the plot label field, the Colour option menu and the Period field and makes the current.

Reset This button resets the Ranges area text fields, the Labels area text fields, the plot label field, the Colour option menu and the Period text field to the current values.

Suggest This button sets the Ranges area text fields, the Labels area text fields, the plot label field, the Colour option menu and the Period text field to the suggested values.

Data This button sets the Ranges area text fields, the Labels area text fields, the plot label field, the Colour option menu and the Period text field to the actual ranges of the data and values from the time series data files.

Get Values This button starts data collection from the plot. It is described in Section 2.10.

4 Installation Notes

4.1 Solaris 2.x Distribution

The MPICTIM 2.10.00 distribution for Solaris 2.3 and later consists of the `mpictim` executable, the `Mpictim` application defaults file, a `README` file and example resources for the `mwm` and `olwm` window managers.

The `Mpictim` application defaults file should normally be installed in the `/usr/openwinhome/lib/app-defaults` directory but some systems may use a different location.

The `mpictim` executable uses GHOST, Motif and Fortran dynamically loaded libraries and requires the GHOST `xgenie` executable and the GHOST fonts. The GHOST environment is available from AEA Technology. The Motif 1.2.3 dynamic libraries are in the `SUNWmfrun` package from the Solaris 2.4 distribution (they can also be installed on Solaris 2.3). The Fortran dynamic libraries

are in the SPROLib77 package which is distributed with SPARCompiler Fortran by SunSoft. (SPARCompiler Fortran 3.0.1 was used to produce the code but the libraries from versions 2.0, 2.0.1 and 3.0 should also work). If the GHOST, Motif and Fortran are installed in the standard locations then `mpictim` should run straight away, otherwise environment variables will need to be set to make the application work.

References

- [1] OSF/Motif Programmer's Guide, Revision 1.1, Open Software Foundation, Prentice Hall, ISBN 0-13-640673-4, 1991.
- [2] N J Brealey, MPICTIM USER'S GUIDE, AEA Technology, Culham Laboratory, AEA/TYKB/31878/TN/10, September 1994.
- [3] W A J Prior, GHOST USER MANUAL Version 8, UKAEA Culham Laboratory, ISBN 0-85311-184-7, 1991.

F CPC 87(1995)155

Reprinted from

Computer Physics Communications

Computer Physics Communications 87 (1995) 155–178

Body-fitted electromagnetic PIC software for use on parallel computers

J.W. Eastwood¹, W. Arter, N.J. Brealey, R.W. Hockney
AEA Technology, Culham Laboratory, Abingdon, Oxfordshire, OX14 3DB, UK

Received 20 September 1994; revised 2 December 1994





ELSEVIER

Computer Physics Communications 87 (1995) 155–178

Computer Physics
Communications

Body-fitted electromagnetic PIC software for use on parallel computers

J.W. Eastwood¹, W. Arter, N.J. Brealey, R.W. Hockney

AEA Technology, Culham Laboratory, Abingdon, Oxfordshire, OX14 3DB, UK

Received 20 September 1994; revised 2 December 1994

Abstract

This paper describes the algorithm and implementation issues for the three-dimensional body-fitted Particle-In-Cell (PIC) software suite 3DPIC for modelling the time evolution of interactions between electromagnetic (EM) fields and flows of relativistic charged particles. 3DPIC is applicable to EM calculations in complex geometries where displacement fields are important, and was developed specifically for microwave device modelling. A description is given of the physical model, the numerical scheme and the software. The software was designed to run efficiently on both serial and distributed-memory parallel computers. The performance achieved by the software on parallel computers demonstrates the potential of this code for large scale time-dependent electromagnetic and electrodynamic calculations.

1. Introduction

The software described in this paper offers powerful new capabilities for electromagnetic PIC [1,2] modelling. It was designed for two-dimensional (2-D) and three-dimensional (3-D) modelling of microwave tubes and microwave transmission where the interaction of electromagnetic waves with charged particle flow is important [4–6], although it can equally well be applied to other time dependent problems normally tackled by finite difference, time domain (FDTD) codes. The software solves the relativistic Maxwell–Vlasov set of equations for the electromagnetic field and particle motion throughout the computational volume; this is appropriate for situations where the characteristic timescales are comparable to the transit time of light across the system. For phenomena on a slower timescale, electrostatic [1] or Darwin models [3] may

be more appropriate. Volume filling lossy dielectrics, magnetic media, conducting grids and surfaces, surface physics and coupling to external circuits can all be incorporated into the simulations.

Particle-in-Cell (PIC) simulation methods have been used to solve the coupled Maxwell–Vlasov system of equations for over two decades. Early simulations (e.g. [7]) were restricted to simple geometries, and repeatedly solved Poisson's equation to correct the longitudinal part of the electric field to obtain charge conservation. Later codes, designed specifically for simulations in non-rectangular domains achieved charge conservation without the solution of Poisson's equation by using a particle integration scheme which moved particles along the principal axes of the grid. Mature codes using this approach include the MRC MAGIC (2-D) [8,9] and SOS (3-D) [10] codes, and the SAIC 3-D code ARGUS [11].

MAGIC, SOS and other codes of similar vintage were designed to exploit the large scientific com-

¹ E-mail: jim.eastwood@aeat.co.uk.

puters of the 1970s. More recent programs, such as QUICKSILVER [12] were developed to exploit shared-memory MIMD computers efficiently. Almost all of the mature codes use finite differences and are restricted to meshes which are unions of rectangular bricks. None are specifically designed to use body fitted coordinates and to exploit distributed-memory MIMD computers.

The new simulation suite 3DPIC differs from established PIC codes [8–12] for microwave modelling primarily in that

- (i) it uses body-fitted finite elements rather than finite differences on an orthogonal grid,
- (ii) the finite element (FE) derivation carries over the simplicity of the Yee FDTD algorithm [13] for the EM calculation to general geometry,
- (iii) the general geometry FE derivation gives current assignment schemes which are charge conserving,
- (iv) it was designed *ab initio* to use distributed-memory parallel computers [14] efficiently.

The last is achieved by a multiblock decomposition of the computational volume. Each block in the decomposition behaves as an independent PIC calculation, which communicates with neighbouring blocks by exchanging fields and particles at its boundaries; these exchanges are treated as message passing, thus allowing simple and efficient mapping of complex physical configurations onto distributed-memory parallel computers.

Blocks are curvilinear hexahedra, i.e. they are cuboid in some (in general non-orthogonal) coordinate system. The elegance of this method is that with appropriate choices of co- and contra-variant tensor components, the majority of the PIC algorithm for each block takes a coordinate invariant form which differs little from the “virtual particle” scheme [15] on regular Cartesian meshes. All geometrical and material properties are encapsulated in the constitutive equations for the fields and in coordinate transformations in the particle equations of motion.

The next section presents the physical model, followed in Section 3 by an outline of the numerical method. Our FE derivation extends the methods introduced by Lewis [16], in two directions: the employment of more general element shapes and the use of the FE approximation for time dependence in the field action integral [15]. The former allow accurate ele-

ment fitting to complex shaped devices, and the latter automatically ensures charge conservation. The importance of exact charge conservation is that it eliminates the need to assign charge and solve Poisson’s equation to correct the longitudinal electric field. Major issues are the description of objects and compact data storage for the computations; these issues are addressed in the Section 4. Implementing the code on parallel computers and benchmarking its performance are treated in Section 5. Section 6 contains concluding remarks.

2. Physical model

2.1. Basic equations

The mathematical model consists of the Maxwell–Vlasov set; Maxwell’s equations for the electromagnetic fields and Vlasov’s equation (possibly relativistic) for the dynamics of the charged particles. The charged particles may be electrons, ions, or both, although the discussion given below is for electrons only. The mathematical model of the physical system is completed by adding descriptions of the emission and absorption of radiation and of particles from boundaries.

If the particle distribution function f is represented by a set of sample points (i.e. “superparticles”)

$$f(\mathbf{x}, \mathbf{p}, t) = \sum_i N_s \delta(\mathbf{x} - \mathbf{x}_i(t)) \delta(\mathbf{p} - \mathbf{p}_i(t)) \quad (1)$$

where $(\mathbf{x}_i, \mathbf{p}_i); i \in [1, N_p]$ are the coordinates of the N_p superparticles, each of mass $M = N_s m_0$ and charge $Q = N_s q$, then the Maxwell–Vlasov set may be written in terms of the action integral

$$I = \int dt d\tau \left(\frac{\mathbf{D} \cdot \mathbf{E} - \mathbf{B} \cdot \mathbf{H}}{2} - \rho \phi + \mathbf{j} \cdot \mathbf{A} \right) - \int \sum_i \frac{M c^2}{\gamma_i} dt \quad (2)$$

where symbols take their usual meanings; \mathbf{A} is the magnetic vector potential, ϕ is the electric potential, $\mathbf{B} = \nabla \times \mathbf{A}$ is magnetic flux, $\mathbf{E} = -\nabla \phi - \dot{\mathbf{A}}$ is electric field, \mathbf{D} is electric displacement, \mathbf{H} is magnetic intensity, ρ is charge density, and \mathbf{j} is current density. γ is the relativistic gamma, and the integrals are over

time t and space τ . Treating I as a functional of the vector potential A , the scalar potential ϕ and particle coordinates $\{x_i\}$ leads to Euler–Lagrange equations which reduce to Maxwell’s equations and the relativistic equations of motion [16,17].

2.2. Material properties

General dielectric and magnetic media may be included in the model by adding polarisation, P , and magnetisation, M , terms to the action integral

$$I = \dots + \int dt d\tau (M \cdot B + P \cdot E) \quad (3)$$

and substituting for M and P from the resulting evolutionary equation using their constitutive relationships. The non-lossy part gives the dielectric function relating D to E , and the lossy parts give the magnetisation and conduction currents, j_l :

$$D = \epsilon E, \quad j_l = -\nabla \times \beta \dot{B} + \sigma E \quad (4)$$

where β is the inductive loss coefficient and σ is the conductivity.

A small magnetisation current can be used in simulations to control numerical noise, thereby allowing simulations to be performed with fewer simulation superparticles than would otherwise be possible.

2.3. Electromagnetic boundary conditions

2.3.1. Internal

Internal boundary conditions are used to treat periodicity, symmetry and to allow domain decomposition. The latter case arises when the computational domain is subdivided into blocks (cf. Section 3.2). Internal boundary conditions are implemented using “glue-patches” (Section 4.1.6).

2.3.2. External

External boundary conditions specify tangential electric fields or normal magnetic fields. They are typically Dirichlet conditions, and may apply at surfaces completely embedded within the computational domain (e.g. an internal conductor).

The full wave description within a device is linked to a lumped circuit approximation of power supplies and extraction waveguides by coupling elements at the

surface of the active computational domain to external circuit elements. The external elements have no physical dimension, but they provide a relationship between surface current, I , and tangential electric fields, E . Thus, a perfect conductor would give $E = 0$, a purely resistive circuit element would give $E = ZI$ and a circuit with resistance and inductance (or capacitance) would give

$$\alpha_c \frac{dE}{dt} + \beta_c E = \gamma_c \frac{dI}{dt} + \delta_c I \quad (5)$$

where circuit coefficients α_c , β_c , γ_c , δ_c are chosen to describe the particular external circuit (see [18]). Similarly, a second order differential equation is used for coupling to an L-R-C circuit and so forth. For steady state circuits, Eq. (5) can be replaced by the complex impedance equation $E = ZI$.

Radiation boundary conditions can be treated using the volume wave absorption method used in MAGIC [8,9]. This involves the addition of a non-physical absorbing medium (giving terms $-\sigma E$ in Ampere’s Law and $-\sigma_B B$ in Faraday’s Law) at free space boundaries.

2.4. Particle boundary conditions

2.4.1. Internal

Internal particle boundary conditions follow the same classification as field boundary conditions, and like their field counterparts are implemented using glue-patches. These boundary conditions conserve particle number and involve transformation of particle position and/or momentum coordinates.

2.4.2. External

External particle boundary conditions specify either particle absorption or emission; for example beam injection, cathode emission and secondary emission. Electron emission from boundary surfaces is treated using standard Monte-Carlo procedures. Beam injection selects superparticle positions and momenta to fit the incoming beam distribution function. Space charge limited emission at cathodes is modelled by introducing sufficient free surface space charge (in the form of superparticles) to bring the normal electric field at the surface to zero. Secondary electrons at boundary surfaces are generated and emitted in response to the

locations and momenta of incident superparticles to fit the chosen energy and direction distribution.

3. Numerical scheme

3.1. Tensor formulation

This section summarises the definitions and identities in general curvilinear coordinates [19] used in the algorithm derivation and formulation.

Let the reference Cartesian coordinates have components (x^1, x^2, x^3) and unit vectors $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$, so that a vector, \mathbf{x} , may be written $\mathbf{x} = x^i \hat{x}_i$, where summation over the repeated index i ($= 1, 2, 3$) is implied. If (x^1, x^2, x^3) are expressed as functions of the curvilinear coordinate components $(\bar{x}^1, \bar{x}^2, \bar{x}^3)$, then $x^1 = x^1(\bar{x}^1, \bar{x}^2, \bar{x}^3)$, etc, or more concisely $\mathbf{x} = \mathbf{x}(\bar{\mathbf{x}})$.

The basis vectors \mathbf{e}_i , and the reciprocal basis vectors \mathbf{e}^i are defined by

$$\mathbf{e}_i = \frac{\partial \mathbf{x}}{\partial \bar{x}^i}, \quad \mathbf{e}^i = \nabla \bar{x}^i \quad (6)$$

These vectors are orthogonal; $\mathbf{e}_i \cdot \mathbf{e}^j = \delta_i^j$ where δ_i^j is the Kronecker delta ($= 1$ if $i = j$, 0 otherwise). The reciprocal basis vectors can be written in terms of the basis vectors $\mathbf{e}^i = \mathbf{e}_j \times \mathbf{e}_k / J$ and vice-versa $\mathbf{e}_i = (\mathbf{e}^j \times \mathbf{e}^k) J$ where the Jacobian $J = \sqrt{g} = |\mathbf{e}_i \cdot \mathbf{e}_j \times \mathbf{e}_k|$ is the square root of the determinant g of the metric tensor.

The (covariant) metric tensor is defined as $g_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j$ and its reciprocal tensor, the contravariant metric tensor is $g^{ij} = \mathbf{e}^i \cdot \mathbf{e}^j$. In Cartesian coordinates, the covariant, contravariant and physical components are identical and the metric tensor reduces to $g_{ij} = \delta_{ij}$. For orthogonal systems, $g_{ij} = 0$ for $i \neq j$. In general g_{ij} is symmetric with six distinct elements, $g_{ij} = g_{ji}$.

3.1.1. Vector identities

A vector \mathbf{A} may be expressed in terms of its contravariant component A^i , covariant components A_i or physical components, $A(i)$:-

$$\mathbf{A} = A^i \mathbf{e}_i = A_i \mathbf{e}^i = A(i) \hat{\mathbf{e}}_i \quad (7)$$

where

$$\hat{\mathbf{e}}_i = \frac{\mathbf{e}_i}{|\mathbf{e}_i|} \quad (8)$$

The permutation symbols ϵ^{ijk} and ϵ_{ijk} are 1 for cyclic indices, -1 for anticyclic indices, and 0 otherwise, and are related to the permutation tensors by

$$\epsilon^{ijk} = \epsilon^{ijk} / \sqrt{g}; \quad \epsilon_{ijk} = \sqrt{g} \epsilon_{ijk} \quad (9)$$

The permutation tensors satisfy the identity

$$\epsilon^{ijk} \epsilon_{klm} = \delta_i^l \delta_j^m - \delta_i^m \delta_j^l \quad (10)$$

Vector dot and cross products become:

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= a^i b_i = a_i b^i \\ (\mathbf{a} \times \mathbf{b})^i &= \epsilon^{ijk} a_j b_k \\ (\mathbf{a} \times \mathbf{b})_i &= \epsilon_{ijk} a^j b^k \end{aligned} \quad (11)$$

The common vector operations are:

$$(\nabla \rho)_i = \frac{\partial \rho}{\partial \bar{x}^i} \quad (12)$$

$$\nabla \cdot \mathbf{a} = \frac{1}{\sqrt{g}} \frac{\partial}{\partial \bar{x}^i} \sqrt{g} a^i \quad (13)$$

$$(\nabla \times \mathbf{a})^i = \epsilon^{ijk} \frac{\partial a_k}{\partial \bar{x}^j} \quad (14)$$

$$\mathbf{a} \cdot \nabla \rho = a^i \frac{\partial}{\partial \bar{x}^i} \rho \quad (15)$$

$$\nabla^2 \rho = \frac{1}{\sqrt{g}} \frac{\partial}{\partial \bar{x}^i} \left(g^{ij} \sqrt{g} \frac{\partial \rho}{\partial \bar{x}^j} \right) \quad (16)$$

3.1.2. Maxwell's equations

Using the formulae of Section 3.1.1 we can write Maxwell's equations in tensor form:-

$$\frac{\partial B^i}{\partial t} = -\epsilon^{ijk} \frac{\partial E_k}{\partial \bar{x}^j} \quad (17)$$

$$\frac{1}{\sqrt{g}} \frac{\partial}{\partial \bar{x}^i} \sqrt{g} B^i = 0 \quad (18)$$

$$\frac{\partial D^i}{\partial t} = \epsilon^{ijk} \frac{\partial H_k}{\partial \bar{x}^j} - j^i \quad (19)$$

$$\frac{1}{\sqrt{g}} \frac{\partial}{\partial \bar{x}^i} \sqrt{g} D^i = \rho \quad (20)$$

It is convenient to introduce extensive current and charge variables

$$I^i = \sqrt{g} j^i; \quad Q = \sqrt{g} \rho \quad (21)$$

and volume scaled flux quantities b^i and d^i for magnetic and displacement fields

$$b^i = \sqrt{g} B^i; \quad d^i = \sqrt{g} D^i \quad (22)$$

If in addition we write the permutation tensor in terms of the permutation symbol, Maxwell's equations become

$$\frac{\partial b^i}{\partial t} = -e^{ijk} \frac{\partial E_k}{\partial \bar{x}^j} \quad (23)$$

$$\frac{\partial b^i}{\partial \bar{x}^i} = 0 \quad (24)$$

$$\frac{\partial d^i}{\partial t} = e^{ijk} \frac{\partial H_k}{\partial \bar{x}^j} - I^i \quad (25)$$

$$\frac{\partial d^i}{\partial \bar{x}^i} = Q \quad (26)$$

Eqs. (23)–(26) have the particularly attractive feature that they have the same form irrespective of coordinate system, and contain no explicit reference to the metrics. This is also true of the relationships between the electromagnetic fields and potentials:

$$E_k = -\frac{\partial \phi}{\partial \bar{x}^k} - \dot{A}_k \quad (27)$$

$$b^i = e^{ijk} \frac{\partial A_k}{\partial \bar{x}^j} \quad (28)$$

The only explicit reference to the metrics appears in the constitutive relationships between fields, which in vacuo are

$$\mu_0 H_i = \frac{g_{ij}}{\sqrt{g}} b^j; \quad \epsilon_0 E_i = \frac{g_{ij}}{\sqrt{g}} d^j \quad (29)$$

More generally, μ_0 and ϵ_0 are replaced by tensor permeabilities and permittivities.

A similar attractive simplicity appears in the Action Integral for the electromagnetic field equations:

$$I = \int dt d\bar{x}^1 d\bar{x}^2 d\bar{x}^3 \times \left(\frac{1}{2} (E_i d^i - H_i b^i) + I^i A_i - Q\phi \right) \quad (30)$$

This metric-free form simplifies the problem of writing a program module for solving Maxwell's equations in arbitrary non-orthogonal coordinate systems.

3.1.3. Equations of motion

The relativistic equations of motion of a charged particle are

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \quad (31)$$

$$\frac{d\mathbf{p}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \quad (32)$$

where

$$\mathbf{p} = \gamma m_0 \mathbf{v} \quad (33)$$

$$\gamma^2 = 1 + \frac{|\mathbf{p}|^2}{(m_0 c)^2} = 1 / \left(1 - \frac{|\mathbf{v}|^2}{c^2} \right) \quad (34)$$

In general coordinates $(\bar{x}^1, \bar{x}^2, \bar{x}^3)$, these become

$$\frac{d\bar{x}^k}{dt} = \bar{v}^k \quad (35)$$

and

$$\frac{d\bar{p}_s}{dt} - \Gamma_{si}^r \bar{p}_r \bar{v}^i = q [E_s + e_{sir} \bar{v}^i b^r] \quad (36)$$

where the Christoffel symbol of the first kind is given by

$$\Gamma_{jk}^i = e^i \cdot \frac{\partial e_k}{\partial \bar{x}^j} \quad (37)$$

The particle dynamics can be included in the action integral, Eq. (30) by adding the extra particle Lagrangian term

$$- \int \frac{m_0 c^2}{\gamma} dt \quad (38)$$

and explicitly expressing the charges and currents in Eq. (30) as sums over particles, i , with charges q_i

$$Q = \sum_i q_i \delta(\bar{x}_i^1 - \bar{x}^1) \delta(\bar{x}_i^2 - \bar{x}^2) \delta(\bar{x}_i^3 - \bar{x}^3) \quad (39)$$

$$I^m = \sum_i q_i \delta(\bar{x}_i^1 - \bar{x}^1) \delta(\bar{x}_i^2 - \bar{x}^2) \delta(\bar{x}_i^3 - \bar{x}^3) \frac{d\bar{x}^m}{dt} \quad (40)$$

The equation of motion Eq. (36) arises from the Euler-Lagrange equation

$$\frac{\partial}{\partial \bar{x}^s} L - \frac{d}{dt} \frac{\partial L}{\partial \bar{v}^s} = 0 \quad (41)$$

where L is the total Lagrangian density.

3.2. Element generation

3.2.1. Multiblock decomposition

The elements are generated using a multiblock decomposition; complex objects are divided into a set of curvilinear hexahedral blocks (where each face has four edges), which in turn are subdivided into hexahedral finite elements using transfinite interpolation. Metric tensors and basis vectors are defined on each element by using coordinate transformations isoparametric to the electric potential representation.

The user of the software controls the multiblock decomposition by describing objects in terms of a “kit-of-parts” (Section 4.2). In selecting the parts for the multiblock division, complex objects are divided into sufficiently convex volumes to avoid the creation of very small, or even negative, volume elements in the blending process; this could cause the numerical simulations to fail. However in the applications envisaged it is unlikely that singular elements will present serious difficulties. Computational efficiency dictates that wherever possible, orthogonal blocks (such as rectangular bricks and cylinder sections) be used as this greatly reduces computational costs and storage. To facilitate load balancing on MIMD computers, it may also be advantageous to subdivide blocks beyond the level dictated by the object's geometry.

Each block of the multiblock decomposition is described by its bounding curves; four intersecting curves in two dimensions and twelve curves in three dimensions. Alternatively, the 3-D block may be described by its bounding surfaces. The relationship of the block coordinates to the global coordinates of the object is described in Section 3.5. The locations of element nodes within a block are generated by blending the interpolants from the bounding curves or surfaces, as described in Sections 3.2.3.

In order to transform between curved space (barred variables) and physical coordinates, and to integrate the equations of motion, values of the basis vectors are required. The integration of the field equations and transformation between covariant and contravariant components requires metric tensor values. These may be evaluated from the finite element approximations to the coordinate transformations, as outlined below.

3.2.2. Isoparametric elements

An isoparametric element uses the same function for the coordinate mapping as for the finite element support of, in this case, the scalar potential ϕ . Fig. 1 shows an isoparametric quadrilateral element in physical and barred coordinate space. A point \mathbf{x} is related to the barred coordinates $\bar{\mathbf{x}} = (\bar{x}^1, \bar{x}^2)$ by

$$\mathbf{x} = \mathbf{x}_1(1 - \bar{x}^1)(1 - \bar{x}^2) + \mathbf{x}_2\bar{x}^1(1 - \bar{x}^2) + \mathbf{x}_3\bar{x}^1\bar{x}^2 + \mathbf{x}_4(1 - \bar{x}^1)\bar{x}^2 \quad (42)$$

This gives a basis vector

$$\mathbf{e}_1 = \frac{\partial \mathbf{x}}{\partial \bar{x}^1} = (\mathbf{x}_2 - \mathbf{x}_1)(1 - \bar{x}^2) + (\mathbf{x}_3 - \mathbf{x}_4)\bar{x}^2 = \mathbf{r}_S(1 - \bar{x}^2) + \mathbf{r}_N\bar{x}^2 \quad (43)$$

and similarly for \mathbf{e}_2 , from which the reciprocal basis vectors, metric tensor elements and Jacobian can be readily computed at any point within the element.

Basis vectors and metrics are defined in the same manner for 3-D hexahedral elements. If we label the nodes of the element by the index triplet (i, j, k) , $i, j, k = 0$ or 1 , and let

$$w_0(z) = (1 - z), \quad w_1(z) = z \quad (44)$$

then the analogue of Eq. (42) is

$$\mathbf{x} = \mathbf{x}_{ijk}w_i(\bar{x}^1)w_j(\bar{x}^2)w_k(\bar{x}^3) \quad (45)$$

and the basis vectors are

$$\mathbf{e}_1 = (\mathbf{x}_{1jk} - \mathbf{x}_{0jk})w_j(\bar{x}^2)w_k(\bar{x}^3) = \mathbf{r}_{\frac{1}{2}jk}w_j(\bar{x}^2)w_k(\bar{x}^3) \quad (46)$$

$$\mathbf{e}_2 = \mathbf{r}_{i\frac{1}{2}k}w_i(\bar{x}^1)w_k(\bar{x}^3) \quad (47)$$

$$\mathbf{e}_3 = \mathbf{r}_{ij\frac{1}{2}}w_i(\bar{x}^1)w_j(\bar{x}^2) \quad (48)$$

The reciprocal basis vectors and metric tensor elements follow by substituting \mathbf{e}_i from Eqs. (45)–(48) into the appropriate formulae from Section 3.1.

3.2.3. Transfinite interpolation

Mesh generation for finite volume and finite element schemes, by blending the interpolants from intersecting pairs of curves such that the mesh exactly fits the boundary curves, was introduced by Gordon and Hall [20] and is now widely used in body-fitted fluid flow

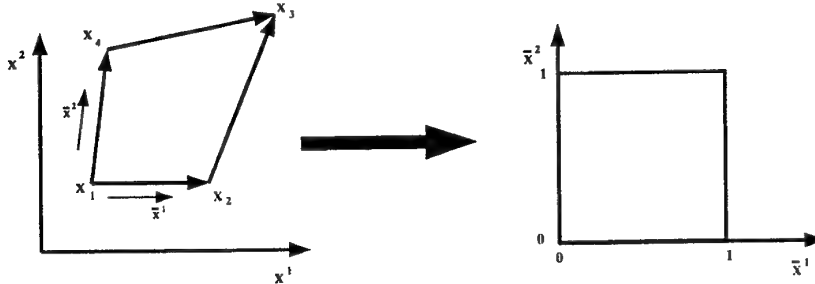


Fig. 1. The isoparametric linear quadrilateral element with corner nodes $x_1 \dots x_4$ maps to the unit square in the barred coordinate space.

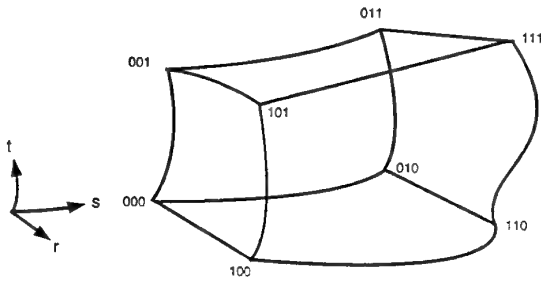


Fig. 2. The six faced volume is defined by the twelve bounding curves joining the corner nodes at positions x_{ijk} , $i, j, k = 0$ or 1

codes, such as Harwell-FLOW3D [21,22]. We summarise here the transfinite interpolation formulae that we use in our body-fitting software.

The aim is to divide the general curvilinear hexahedra as illustrated in Fig. 2 into a set of hexahedral elements which map to unit cubes in curved space. The bounding curve $X_{ij}(r)$, $r \in [0, 1]$ joins the corner node x_{0ij} at $r = 0$ to node x_{1ij} at $r = 1$. Similarly $Y_{ij}(s)$ joins x_{i0j} to x_{i1j} and $Z_{ij}(t)$ joins x_{ij0} to x_{ij1} , where $i, j = 0$ or 1 . The interpolation functions for the r , s , and t coordinates are respectively λ , ψ and η .

Blending interpolants between the bounding curves to obtain exact fits at all eight edges gives the formula

$$\begin{aligned} x(r, s, t) = & X_{jk}(r)\psi_j(s)\eta_k(t) \\ & + Y_{ik}(s)\lambda_i(r)\eta_k(t) \\ & + Z_{ij}(t)\lambda_i(r)\psi_j(s) \\ & - 2x_{ijk}\lambda_i(r)\psi_j(s)\eta_k(t) \end{aligned} \quad (49)$$

where sums over $i, j, k = 0$ and 1 are implied. Gener-

ally, the functions λ , ψ and η are chosen to be linear.

3.3. Maxwell's equations

The discrete finite element approximations to Maxwell's equations are obtained, following Refs. [4,15], using virtual particle electromagnetic particle-mesh schemes for general 3-D coordinate systems. The present implementation uses the lowest order conforming elements, and employs lumping to maintain explicit time integration. If more accurate wave dispersion is required, then the lumping approximations could be replaced by "mass" matrix inversions.

Discrete equations are derived by introducing finite element approximations to the potentials

$$\phi = \Phi U \quad (50)$$

$$A_i = A_{(i)} W_{(i)} \quad (51)$$

where Φ and $A_{(i)}$ are nodal amplitudes. Eq. (50) is shorthand (Eq. (51) similarly) for

$$\begin{aligned} \phi(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) \\ = \sum \Phi(k, n) U(k, n; \bar{x}^1, \bar{x}^2, \bar{x}^3, t) \end{aligned} \quad (52)$$

where the sum is over spatial (k) and temporal (n) node indices. The approximations (50) and (51) are substituted into the Action Integral Eq. (30), which is then varied with respect to the nodal amplitudes Φ and A to yield discrete approximations to the inhomogeneous Maxwell's Eqs. (25) and (26). (The homogeneous Maxwell Equations (23) and (24) follow by virtue of Eqs. (27) and (28), if U and W_i are appropriately chosen.)

3.3.1. Finite element field approximations

The lowest order conforming choice of element support for the potentials is a mixture of piecewise linear and piecewise constant functions. Eqs. (27) and (28) are satisfied in a strong sense for the lowest order conforming elements provided that the finite element approximations to the fields are

$$d^i = d^{(i)} V_{(i)} \quad (53)$$

$$E_i = E_{(i)} V_{(i)} \quad (54)$$

$$b^i = b^{(i)} X_{(i)} \quad (55)$$

$$H = H_{(i)} X_{(i)} \quad (56)$$

where

$$W_i(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = \Pi(\bar{x}^i) \Lambda(\bar{x}^j) \Lambda(\bar{x}^k) \Lambda(t) \quad (57)$$

$$U(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = \Lambda(\bar{x}^1) \Lambda(\bar{x}^2) \Lambda(\bar{x}^3) \Pi(t) \quad (58)$$

$$V_i(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = \Pi(\bar{x}^i) \Lambda(\bar{x}^j) \Lambda(\bar{x}^k) \Pi(t) \quad (59)$$

$$X_i(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = \Lambda(\bar{x}^i) \Pi(\bar{x}^j) \Pi(\bar{x}^k) \Lambda(t) \quad (60)$$

$$\Lambda(\bar{x}^i) = \begin{cases} 1 - |\bar{x}^i| & ; |\bar{x}^i| < 1 \\ 0 & ; \text{otherwise} \end{cases} \quad (61)$$

$$\Pi(\bar{x}^i) = \begin{cases} 1 & ; -1/2 < \bar{x}^i \leq 1/2 \\ 0 & ; \text{otherwise} \end{cases} \quad (62)$$

and (i, j, k) are cyclic permutations of indices $(1, 2, 3)$. Fig. 3 shows the location of the nodes at which amplitudes of these finite element field approximations are defined. The following two subsections discuss how this choice of FE approximation yields field equations of a similar form to the familiar staggered leapfrog finite difference equations [13] and charge assignment similar to the standard CIC scheme [1,2]. Higher order schemes could be generated by using higher order FE support functions, but are not considered further here.

3.3.2. Homogeneous equations

The homogeneous equations follow immediately from the definitions of the finite element fields and Eqs. (23), (24), (27) and (28). Magnetic flux is identically conserved and Faraday's law

$$\frac{\partial b^i}{\partial t} = -e^{ijk} \frac{\partial E_k}{\partial \bar{x}^j} \quad (63)$$

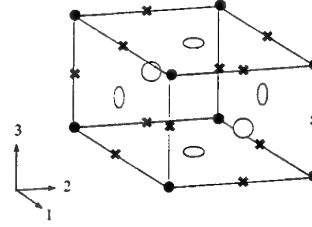


Fig. 3. The location of nodes on the unit cube element. Crosses give A_i , E_i , I^i and d^i node locations, open circles give H_i and b^i locations and solid circles give element corner and Φ node locations.

gives

$$b^i \frac{\partial X_{(i)}}{\partial t} = -e^{ijk} E_k \frac{\partial V_{(k)}}{\partial \bar{x}^j} \quad (64)$$

Using the above definitions of X_i and V_i shows that Eq. (64) reduces to the standard Yee [13] finite difference form of Faraday's law, but for nodal amplitudes of field components b^i and E_k .

3.3.3. Inhomogeneous equations

Ampere's equation is obtained by assembling contributions to the variation of the Action from adjacent elements

$$\frac{\delta I}{\delta A_i} = \int dt d\bar{x}^1 d\bar{x}^2 d\bar{x}^3 \left\{ -d^i \frac{\partial W_{(i)}}{\partial t} V_{(i)} - H_j e^{jlm(i)} \frac{\partial W_{(i)}}{\partial \bar{x}^m} X_{(j)} + I^{(i)} W_{(i)} \right\} = 0 \quad (65)$$

The current term, which gives the prescription for assigning current from particles to the elements, is discussed further in Section 3.4.1.

For each node internal to a block, there are eight contributions (from four adjacent elements at two timelevels) to the assembled equation resulting from Eq. (65) for a displacement field component. At block surfaces, this number may be different. In the implementation of the software, element contributions are pre-assembled for nodes internal to blocks, but contributions across block faces are assembled by "glue-patch" data exchanges (cf. Section 4.1.6) only during the computation of the displacement fields. For surface nodes, there are contributions from elements within the computational volume, plus possibly

surface current terms from the external boundary conditions.

Variation of I with respect to the potential nodal amplitudes gives the discrete approximations to Gauss' Law:

$$\frac{\delta I}{\delta \Phi} = \int dt d\bar{x}^1 d\bar{x}^2 d\bar{x}^3 \left\{ -d^i \frac{\partial U}{\partial \bar{x}^i} V_{(i)} - qU \right\} = 0 \quad (66)$$

The last term gives the prescription for charge assignment, and the remaining part of the integral gives contributions to the divergence of the displacement field. As for Ampere's equations, the number of contributions to a node at a block surface may differ from that for nodes interior to blocks, and boundary conditions are applied via surface patches.

For simplicity, the present implementation of the software uses the lumped approximations $\langle V_{(i)}, V_{(i)} \rangle = 1$, $\langle X_{(i)}, X_{(i)} \rangle = 1$, where the inner product notation

$$\langle a, b \rangle = \int_D dt d\bar{x}^1 d\bar{x}^2 d\bar{x}^3 a(\bar{x}^i, t) b(\bar{x}^i, t) \quad (67)$$

is used for the integral over the domain of interest D . The advantage of lumping is that it greatly reduces the amount of computational work per node per timestep, but at the cost of degrading dispersive properties and introducing the possibility of numerical Cerenkov instability [23]. With lumping, the assembled equation resulting from Eq. (65) becomes

$$\partial_t d^i = e^{ijk} \partial_j H_k - I^i \quad (68)$$

where the operators ∂ denote centred difference. Lumping in time has the advantage of making timestepping explicit.

3.3.4. Constitutive relations

For isotropic permeability and permittivity tensors the finite element approximations to the constitutive relations are

$$\langle E_i V_{(i)}, V_{(i)} \rangle = \left\langle \frac{1}{\epsilon \sqrt{g}} g_{ij} d^j V_{(j)}, V_{(i)} \right\rangle \quad (69)$$

and

$$\langle H_i X_{(i)}, X_{(i)} \rangle = \left\langle \frac{1}{\mu \sqrt{g}} g_{ij} b^j X_{(j)}, X_{(i)} \right\rangle \quad (70)$$

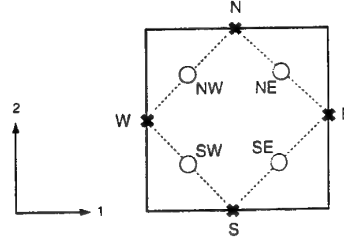


Fig. 4. Values of G_{12} at the NW, NE, SW and SE nodes couple values of d^1 , E_1 , and the N and S nodes to d^2 , E_2 at the E and W nodes.

Using lumped approximations these reduce to

$$E_i = G_{ij}^E d^j \quad (71)$$

$$H_i = G_{ij}^H b^j \quad (72)$$

Elements of the symmetric tensors G_{ij}^E and G_{ij}^H are sparse matrices. More general permittivities and permeabilities are handled by replacing ϵ and μ by tensor functions in the computation of G_{ij}^E and G_{ij}^H , respectively.

The constitutive relations (71) and (72) resulting after assembly at an internal node generally have four off-diagonal terms for each element. Fig. 4 illustrates the location of the nodes. For example, contributions from the element to E_1 at the W node are

$$\delta E_1 = G_{11}^E(W) d^1(W) + G_{12}^E(NW) d^2(N) + G_{12}^E(SW) d^2(S) \quad (73)$$

and to H_2 at the W node

$$\delta H_2 = G_{22}^H(W) b^2(W) + G_{12}^H(NW) b^1(N) + G_{12}^H(SW) b^1(S) \quad (74)$$

Similar expressions hold for other components at the other nodes. Adding contributions of all elements to a given node implies a four point sum for the off-diagonal terms.

3.3.5. Metric computations

The metric terms are defined by (no implied summation)

$$G_{ij}^E = \left\langle \frac{g_{ij}}{\epsilon \sqrt{g}} V_j, V_i \right\rangle \quad (75)$$

and

$$G_{ij}^H = \left\langle \frac{g_{ij}}{\mu\sqrt{g}} X_j, X_i \right\rangle \quad (76)$$

The integrals in Eqs. (75) and (76) are evaluated element-by-element in the preprocessor (Section 4.2.1) using Gaussian quadrature. This metric computation assembles the terms within the interior of a uniblock, but leaves the surface coefficients unassembled to facilitate patch exchange.

3.3.6. Electromagnetic boundary conditions

The internal electromagnetic boundary conditions outlined in Section 2.3 are dealt with by glue-patch exchange (Section 4.1.6), and the external boundary conditions described in this section appear as extra terms in the evolutionary equations for fields at surface nodes.

For a surface of a block arranged so that its normal is parallel to e^3 , the vector equivalent of Ohm's Law $ZI = E$ is

$$e^3 \times (ZI_s - E) = 0 \quad (77)$$

where the surface current I_s has $I_s^3 = 0$. Now for an arbitrary vector f , straightforward manipulation gives

$$\begin{aligned} e^3 \times f = & \sqrt{g} \{ e^1 (-g^{33} f^2 + g^{23} f^3) \\ & + e^2 (g^{33} f^1 - g^{13} f^3) \\ & + e^3 (-g^{23} f^1 + g^{13} f^2) \} \end{aligned} \quad (78)$$

Taking

$$f = ZI_s - E$$

and noting that

$$f^3 = -E^3 = \frac{-d^3}{\epsilon\sqrt{g}} \quad (79)$$

it follows, from equating the components of Eq. (78) separately to zero that

$$I_s^i = \frac{1}{\epsilon Z \sqrt{g}} (d^i - \frac{g^{i3} d^3}{g^{33}}), \quad i = 1, 2 \quad (80)$$

Geometrical considerations lead to the result that the volume current is given by

$$I = I_s | e_1 \times e_2 | \quad (81)$$

Algorithmically, the first term on the right-hand side of Eq. (80) presents few problems. The second vanishes if the 3-coordinate is normal to the surface in the sense that $g_{13} = g_{23} = 0$, since $g^{ij} g_{j3} = 0$ for $i = 1, 2$. Alternatively, if the coordinates within the surface are orthogonal so that $g_{12} = 0$, then the relations

$$\frac{g^{i3}}{g^{33}} = -\frac{g_{i3}}{g_{(i)(i)}}, \quad i = 1, 2 \quad (82)$$

apply and $g^{i3} d^3 / g^{33}$ can be replaced by

$$S_i = -\frac{G_{i3}^E d^3}{G_{(i)(i)}^E} \quad (83)$$

The d^i on a resistive wall patch are thus updated using

$$d^{i(n+1)} = \alpha d^{i(n)} + \beta \dot{d}^{i(n)} + \gamma_i, \quad i = 1, 2 \quad (84)$$

where n and $n+1$ denote time levels. Representing d^i in Eq. (80) as $\theta d^{i(n+1)} + (1-\theta) d^{i(n)}$ it follows that

$$\alpha = \frac{\tilde{Z} + 2\theta - 2}{\tilde{Z} + 2\theta} \quad (85)$$

$$\beta = \frac{2\tilde{Z}}{\tilde{Z} + 2\theta} \quad (86)$$

and

$$\gamma_i = \frac{2S_i}{\tilde{Z} + 2\theta} \quad (87)$$

where

$$\tilde{Z} = \frac{\epsilon Z}{\Delta t \sqrt{g^{33}}} \quad (88)$$

Note that provided $\theta > 1/2$ the above can be used to represent a perfectly conducting wall. Boundary conditions involving the specification of normal magnetic field and tangential electric field are straightforwardly implemented; the latter are in many cases equivalent to specifying a potential difference across a block.

3.3.7. Timestep loop

The timestep loop for the integration of the field equations deals exclusively with field components. Input from the particle integration are the contravariant currents I^i , and the output to the particle integration are field components E_i and b^i . Details of the particle integration and boundary condition will be discussed later. The finite element field equations are assembled

within each block, and glue-patch data exchange is used to complete the assembly at block surfaces. The electromagnetic field integration routines advance the primary field arrays - the set of nodal amplitudes of d^i and b^i - by the electromagnetic integration timestep (\leq particle timestep). This leads to the following operations in the timestep loop, starting from currents l^i and fields b^i at time level $n + 1/2$ and displacement fields at time level n :

- (i) obtain H_i from b^i in each block (Eq. (72)).
- (ii) compute the contributions \dot{d}^i to the displacement current in each block, using the right hand side of Eq. (68) for internal nodes, and its partially assembled equivalent for surface nodes.
- (iii) assemble \dot{d}^i contributions at block surfaces by glue-patch data exchange. This step applies the "internal" boundary conditions (of periodicity, symmetry, Neumann type and domain decomposition).
- (iv) update d^i and apply external boundary conditions. At internal boundaries, the new d^i is computed from $\partial_t d^i = \dot{d}^i$. The treatment of external nodes is described in Section 3.3.6.
- (v) obtain E_i from d^i in each block (Eq. (71)).
- (vi) assemble E_i contributions across internal boundaries by glue-patch data exchange.
- (vii) apply external boundary conditions to E_i .
- (viii) advance b^i in each block (Eq. (64)).

At the end of the electromagnetic timestep, nodal values d^i are known at time level $n + 1$ and b^i are known at time level $n + 3/2$.

3.4. Particle equations

The electromagnetic field equations take a particularly simple form when the appropriate choices of co- and contra-variant fields are made; the same is true for current assignment, force interpolation and the integration of the particle equations of motion. The only substantial difference between the algorithm in general geometry and in Cartesian coordinates arises in the update of particle positions.

3.4.1. Assignment

The approximation of representing the distribution function by a set of "superparticle" sample points reduces the velocity space integrals for charge and current density to sums over particles (Eqs. (39) and

(40)). Substituting these sums into the source term integrals arising from the variations of the field Lagrangian with respect to Φ and A_i give respectively the prescriptions for charge assignment

$$Q = \int dt \sum_p q_p U(\bar{x}_p^1, \bar{x}_p^2, \bar{x}_p^3, t) \quad (89)$$

and for current assignment.

$$l^i = \int dt \sum_p q_p \dot{\bar{x}}_p^{(i)} W_{(i)}(\bar{x}_p^1, \bar{x}_p^2, \bar{x}_p^3, t) \quad (90)$$

The integrals for Q and l^i are evaluated in exactly the same manner as described in Ref. [15]. Motion is assumed to be impulsive and trajectory segments are assumed to be straight lines in the curved $(\bar{x}^1, \bar{x}^2, \bar{x}^3)$ space in which elements are unit cubes. The sums in Eqs. (89) and (90) are over particles p . Since the expressions are linear, there is no loss of generality in considering a single particle of charge q .

Charge assignment. If we label nodes on the unit cube element in curved space (i, j, k) as shown in Fig. 5 and write the assignment function in terms of its product parts

$$U_{ijk}(\bar{x}^1, \bar{x}^2, \bar{x}^3, t) = w_i(\bar{x}^1) w_j(\bar{x}^2) w_k(\bar{x}^3) \quad (91)$$

the charge assigned to element node (i, j, k) where $i, j, k = 0$ or 1 becomes

$$Q_{ijk} = q w_i(\bar{x}^1) w_j(\bar{x}^2) w_k(\bar{x}^3) \quad (92)$$

where $w_0(x) = 1 - x$, $w_1(x) = x$ and $(\bar{x}^1, \bar{x}^2, \bar{x}^3)$ is the displacement of the particle position from the element corner $(0, 0, 0)$. Inspection of Eq. (92) reveals the assignment scheme to be the same as area weighting [1,2], but now applied on a uniform net in curved space rather than in Cartesian, physical space. Locating the element containing the particle is trivial, since the element index is given by the integer truncation of the particle coordinates \bar{x}^i , exactly as for simple Cartesian PIC.

Current assignment. Using the impulse approximation to particle motion as described in [15] and the product form of W reduces Eq. (90) for the contribution of a single particle to

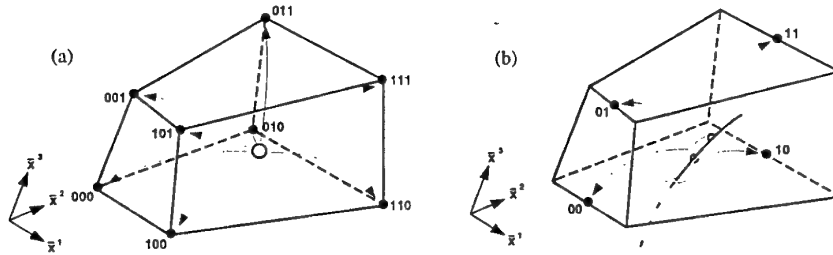


Fig. 5. (a) Charge assignment from a single particle and (b) l^1 current assignment from a single particle trajectory segment for linear potential test functions in physical space.

$$l_{\frac{1}{2}jk}^1 = \int_{-1/2}^{1/2} ds qs \Delta \bar{x}^1 w_j(\bar{X}^2 + s \Delta \bar{x}^2) w_k(\bar{X}^3 + s \Delta \bar{x}^3) \quad (93)$$

The quadratic integral is evaluated exactly by point Gaussian quadrature, so Eq. (93) can be written as the contribution from “virtual particles” at \bar{x}_{\pm} :

$$l_{\frac{1}{2}jk}^1 = q \frac{\Delta \bar{x}^1}{2} [w_j(\bar{x}_+^2) w_k(\bar{x}_+^3) + w_j(\bar{x}_-^2) w_k(\bar{x}_-^3)] \quad (94)$$

where indices j and k take values 0 or 1. The positions of the virtual particle have coordinates

$$\bar{x}_{\pm}^k = \bar{X}^k \pm \frac{\Delta \bar{x}^k}{2\sqrt{3}} \quad (95)$$

where $\bar{X}^k = (\bar{x}_f^k + \bar{x}_i^k)/2$, $\Delta \bar{x}^k = \bar{x}_f^k - \bar{x}_i^k$, and $k = 1, 2$ or 3 . Coordinates $(\bar{x}_i^k, \bar{x}_f^k)$ and $(\bar{x}_f^k, \bar{x}_i^k)$ are respectively the start (i) and end (f) of the trajectory segment. The corresponding expressions for current components $l_{\frac{1}{2}jk}^2$ and $l_{\frac{1}{2}jk}^3$ are given by simultaneously cyclically permuting component indices (123) and node indices $(\frac{1}{2}jk)$ in Eq. (94). Fig. 5 gives a Cartesian space sketch of assignment according according to (a) Eq. (92) for charge and (b) Eq. (94) for current.

In practice, it more convenient to evaluate explicitly the integrals for l^i rather than use quadrature. Eq. (94) then becomes

$$l_{\frac{1}{2}jk}^1 = q \Delta \bar{x}^1 \left[w_j(\bar{X}^2) w_k(\bar{X}^3) + (j - \frac{1}{2})(k - \frac{1}{2}) \frac{\Delta \bar{x}^2 \Delta \bar{x}^3}{3} \right] \quad (96)$$

and similarly for l^2 and l^3 by cyclic permutation of indices.

3.4.2. Charge conservation

By linear superposition, conservation for one trajectory moving through a single time step implies the same for the sum of all trajectories. Summing all contributions to Eqs. (93) from a single particle, gives the same as the difference of Eqs. (92) at start and end points, i.e. the linear hexahedral element case satisfies

$$\partial_i Q = -\partial_k l^k \quad (97)$$

where the symbol ∂ denotes a centred difference arising from assembling the finite element contributions, and Q and l^k are nodal amplitudes. Equations of the form (97) can be shown to be generally satisfied for virtual particle algorithms.

The great benefit of Eq. (97) is computational speed-up; charge assignment and solution of Poisson's equation for the electrostatic potential correction are not needed in the timestep loop.

3.4.3. Force interpolation

Fields at particle positions are given by the finite element fields (Eqs. (53)–(56)) sampled at the particle positions, where the interpolation functions are given by Eqs. (57)–(62).

3.4.4. Momentum equation

Particle momenta and positions are stored in terms of their curvilinear coordinates local to the block which contains them. The momentum coordinates are updated in two stages. First, the momentum components are updated at the old particle position, then are transformed to components measured at the new

position; this eliminates the explicit appearance of the Christoffel symbols terms in the discrete approximation to Eq (36).

The established centred time approximation [7,1,2] is used for the momentum update at the old particle position. Covariant momentum components k at time level $n+1/2$, $\bar{p}_k^{(n+1/2)}$, measured at the current particle positions $\bar{x}^{k(n)}$ are found using

$$\begin{aligned} \bar{p}_k^{(n+1/2)} - \bar{p}_k^{(n-1/2)} \\ = q\Delta t \left(E_k^{(n)} + e_{klm}(\bar{v}^{l(n-1/2)} \right. \\ \left. + \bar{v}^{l(n+1/2)})b^{m(n)}/2 \right) \end{aligned} \quad (98)$$

where $\bar{v}^l = \bar{p}^l/\gamma m_0$ are contravariant velocity components and e_{klm} is the permutation symbol. Eq. (98) is solved using the method due to Boris [7]; apply a half electric acceleration, compute the relativistic γ , apply a two stage rotate and then complete the electric acceleration. Fields are evaluated at the particle positions using Eqs. (54) and (55).

3.4.5. Position equation

Positions are updated to new time level $n+1$ and the momentum components at the new positions are computed using a predictor-corrector scheme to solve

$$\begin{aligned} \bar{x}^{k(n+1)} - \bar{x}^{k(n)} \\ = \left(\bar{v}^{k(n+1/2)}(\bar{x}^{(n)}) + \bar{v}^{k(n+1/2)}(\bar{x}^{(n+1)}) \right) \Delta t/2 \end{aligned} \quad (99)$$

and

$$\begin{aligned} \bar{p}^{l(n+1/2)}(\bar{x}^{(n+1)}) \\ = e^l(\bar{x}^{(n+1)}) \cdot e_k(\bar{x}^{(n)}) \bar{p}^{k(n+1/2)}(\bar{x}^{(n)}) \end{aligned} \quad (100)$$

In Cartesians, this scheme reduces to the usual Lorentz force leapfrog integration scheme and no iteration is required. In cylindrical coordinates, the predictor-corrector can be replaced by direct matrix inversion in the manner proposed by Boris [7]. Both Cartesian and curvilinear coordinates have been tried for the particle integration, and for orthogonal and near orthogonal meshes they give comparable results. The curvilinear option outlined here is favoured because it requires considerably less work per particle. However, its accuracy on distorted elements has not yet been examined.

3.4.6. Particle boundary conditions

Coordinates of particles leaving a block are collected and sorted into tables of particle passing through given glue-patches or boundary condition patches. Internal boundary conditions are applied at the glue-patches by transforming the particle coordinates to those used in the target block. External boundary conditions are applied by deleting outgoing particles and adding incoming particles according to the surface particle source model (e.g. beam, space charge limited emitter, secondary emitter, etc).

3.5. Coordinate transformations

The coordinate systems used in the software are:

- (i) global Cartesians
- (ii) block Cartesians
- (iii) block orthogonal
- (iv) block contravariant
- (v) block covariant

Global positioning information comprises two parts: physical position and logical position. The physical position of a block is described in terms of its origin x_{000} and its rotation matrix R . Global physical positioning is used to graphically display the data. Logical position, described by the glue-patches joining blocks together, is used to transform curvilinear field and particle coordinate components between blocks. This is described in Section 4.1.6.

In most cases, the local Cartesian and local orthogonal coordinates are the same. However, for special cases (e.g. polar mesh segment block), it becomes advantageous for metric data compression to store the basis vectors e_i in terms of components of a reference orthogonal system which is not Cartesian. When local orthogonal coordinates are not Cartesian, their scale factors need to be stored.

The basis vectors define the transformations between orthogonal coordinate components and curvilinear, and the covariant / contravariant transformations use the metric tensor. The orthogonal components are used only for input and output.

4. Software

The management of complexity has been the major challenge in developing the PIC software suite 3DPIC;

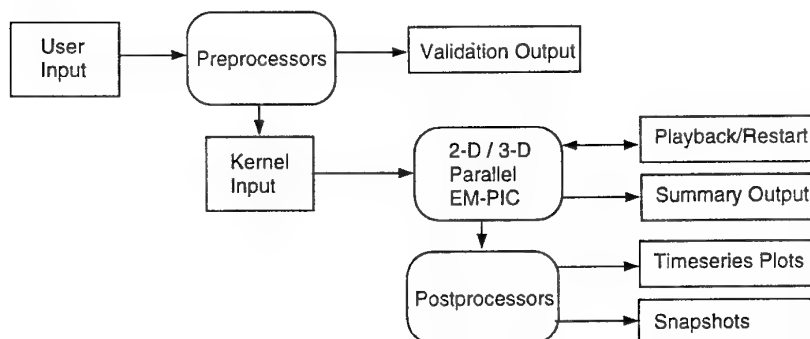


Fig. 6. A schematic of the division of the 3-D electromagnetic PIC software suite 3DPIC into preprocessor, simulation and postprocessor program units.

to produce a general geometry body-fitted code that is both portable and efficiently utilises multiprocessor distributed-memory parallel computers has required careful hierarchical structuring of both data and program units.

The first division is into the interactive (and serial) programs for preprocessing (PEGGIE / P3DTWT) and postprocessing (PICTIM / PICVIS), and into the simulation program (PIC3D) which runs non-interactively on the parallel computer. This is shown schematically in Fig. 6. The preprocessors take compact descriptions of the device geometry and material properties, the external circuit boundary conditions, the initial values for the simulation and the selections of output datasets to be generated and saved. They validate the input for consistency, and produce initialisation summaries and input datasets for the kernel simulation program. Additionally, data may come from restart files from previous runs and from playback files (e.g. beam data) from other software. The preprocessors exchange some flexibility for simplicity, so the kernel input dataset is in human readable form to allow the insertion of special conditions not catered for by the preprocessors. The simulation code PIC3D produces summary output and binary datasets for the postprocessors. These postprocessors take the time series and snapshot datasets and analyse them to produce data in the form suitable for human interpretation. The preprocessors, simulation program and postprocessors share a number of program units, and this is catered for by grouping the program units into a set of libraries. A discussion of the multiblock de-

composition is presented in the next section, followed by a brief description of the preprocessors and postprocessors in Sections 4.2 and 4.3.

4.1. Multiblock decomposition

The multiblock decomposition of the computational domain into a set of curvilinear hexahedral bricks (uniblocks), where each uniblock has its own fields and particles and communicates with other blocks and boundaries via its glue-patches, provides our route to efficiency, portability and flexibility. The resulting numerical algorithm within each block is almost as simple as that for Cartesian PIC in a rectangular brick. The regular (i, j, k) addressing within each block allows data compression of basis vector and metric storage (Section 4.1.5).

Fig. 7 uses the simple case of a cylinder to illustrate the multiblock decomposition of a complex object into a set of curvilinear uniblocks connected by glue-patches. The cylinder (a) is divided into five blocks (b). These blocks are treated as separate objects with their own coordinate systems, and communicate by passing field and particle data via the glue-patches (denoted by dashed lines in (c)). In (d) the curvilinear quadrilateral (hexahedral in 3-D) blocks are meshed by transforming them to rectangles (bricks in 3-D) in curved space and dividing the rectangles (bricks) into unit side square (cube) elements.

The simulation program treats a device as a collection of uniblocks connected together by a network of glue-patches. Each glue-patch sticks together two ad-

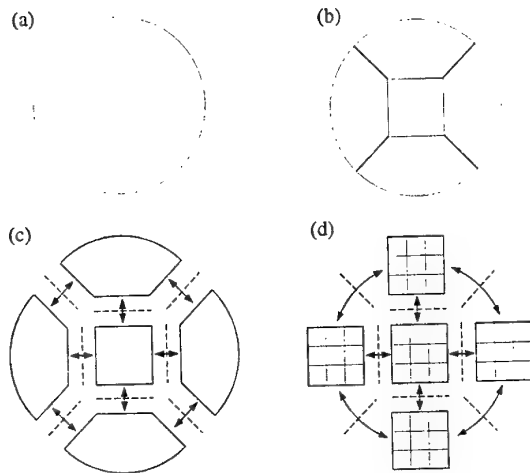


Fig. 7. An illustration of the multiblock decomposition and meshing of a cylinder.

jacent block faces (or block edges in the case of line patches). Different devices are constructed by glueing different shaped multiblocks together and by changing boundary condition patches at the uniblock surfaces. Different material types are treated by changing the data associated with the block type.

Uniblocks map naturally onto processes on a MIMD computer architecture; the multiblock decomposition may be thought of as setting up a set of independent PIC calculations on each processor, with the (time dependent) particle and field boundary conditions being provided by messages from the other processors. All the field and particle data for a given block is local, so the only data that must be exchanged for the calculation to proceed is the boundary data.

To handle complex shapes, some uniblocks will need to be small, whereas for simple shapes, physical boundary condition constraints allow large blocks. If desired, large blocks can be subdivided to facilitate mapping onto the MIMD computer. The data organisation allows several small blocks to be collected together on a given process to obtain effective load balance and allows uniblock data to be moved from one process to another.

Without compromising the subdivision of the spatial mesh into blocks to get efficient MIMD processing, one can further demand that boundary conditions

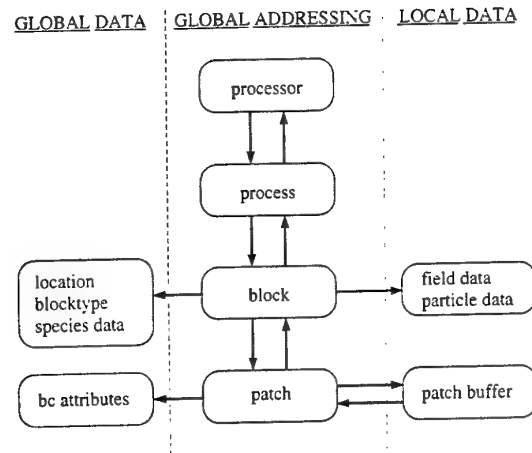


Fig. 8. The data in the main simulation program PIC3D is divided into global addressing, global data and local data.

only apply at the surfaces of blocks². This completely eliminates the addressing problems caused by embedding surfaces within blocks, and allows surface data to be passed to the control program through the glue-patch tables. When a large number of small blocks is needed to describe a complex object, load balancing is achieved by assigning several blocks to one processor.

4.1.1. Global addressing

The global addressing in PIC3D describes the distribution of blocks over the processors of the MIMD computer, and the logical connection of the blocks; this addressing is used to pass information from the surface of one block on one processor to the surface of another block which may be on another processor. The complete global addressing network can be described by the positioning of the two faces of each glue-patch on the uniblocks. The four levels in this network, the processor, process, block and patch are shown schematically in Fig. 8.

Processor: To aid portability a distinction is made between the logical processes used in the program and the physical processors of the MIMD computer; tables are set up to provide the mapping between processes

²This condition is not strictly applied in the program, as it was found advantageous to allow special blocks with embedded structures to describe the geometry of helical travelling wave tubes.

and processors. The simulation program is written in terms of processes, which are translated to physical processor numbers by these tables. When the software is run on single processor workstations, there is simply one process and one processor.

Process: Each process corresponds to a running (master or slave) copy of the simulation program, and may have one or more blocks assigned to it. This allows the computational load to be balanced by moving blocks between processes.

Block: To each block corresponds global block data, namely the block's location in space relative to some global coordinates, its type and global attributes of the particles within it. The block type data contains metric information needed by the field solver and by the particle integrator.

Patch: All exchange of data between blocks is performed through the patch exchange subroutine. This recognises whether a target patch to which it is to send data is on a block belonging to its process, or on a block being computed by another process. In the former case, a memory-to-memory copy of the data is performed, and in the latter a message is sent to the target process.

Addressing in the direction of the downward arrows in Fig. 8 (processor \rightarrow process \rightarrow block \rightarrow patch) uses ordered tables, and the reverse connections use linked lists.

4.1.2. Global data

Physical and numerical data which spans many blocks is stored as global data; examples of this are particle species data, boundary condition data, block type data (e.g. metrics) and global Cartesian positions of the blocks.

4.1.3. Local addressing

Local addressing manages the storage of the principal field and particle arrays, and the transfer of field and particle data to and from the patch buffers used in the glue-patch exchanges. One-dimensional addressing is used for the principal field and particle arrays to allow the same compact coding to be used for both 2-D and 3-D computations. Linked lists are used to sort particles into the glue-patch buffers.

4.1.4. Local data

The principal local data arrays are those for nodal amplitudes l^i , d^i and b^i , for the particle coordinates (\bar{x}^i, \bar{p}^i) , a workspace array to hold alternately E_i and H_i , and input / output buffers for the glue-patch exchanges. On a given process, the nodal amplitude data for blocks are stored as a sequence of fixed length records in the appropriate field or current array. Particle and buffer data use variable length records.

4.1.5. Memory usage

PIC codes are always large consumers of computer memory, and the extra data needed to describe general geometry can greatly inflate this need. This has been avoided in PIC3D by using data compression for metric and basis vector storage, and dynamic memory allocation for the large local data arrays.

In most cases, symmetries and congruences greatly reduce the amount of metric data to be stored. For instance, the example shown in Fig. 7 extended to 3-D perpendicular to the plane shown has two block types only; the central rectangular brick block type, which requires only six numbers to completely specify its basis vectors and metric tensor elements, and the surrounding wedge shaped block type, which requires a single plane of values for metrics. The metric data compression for this case has reduced the memory requirement from the full metric storage for the block $\sim 75N^3$ to $\sim 7 + 3N^2$, assuming an $N \times N \times N$ mesh in each block. These savings are typical, and similarly apply to the basis vector storage. Physical coordinates of element nodes are not stored, but are constructed when needed (for diagnostic output only).

The preprocessors compute the amount of memory required for the field arrays and estimate how much is needed for particle coordinates and patch exchange buffers; the simulation code allocates this space when it starts up.

4.1.6. Patch exchange

Patch exchanges are used to transfer field and particle data between blocks. Field glue-patch exchanges correspond to the transfer of fixed length records, whereas the length of records for particle exchange varies depending on the particle flux.

Field glue-patches. For all cases, the application of interior field boundary conditions involves assembling

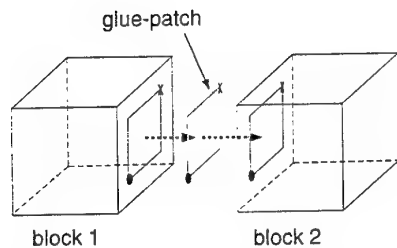


Fig. 9. All interior boundary conditions are applied by passing data between blocks via glue-patches.

contributions to d or E across adjacent block boundaries. This assembly is performed by adding the glue-patch contributions. Fig. 9 illustrates the glue-patch data transfer. Data stored on the logical (i, j, k) lattice of nodes in block 1 are copied to the glue-patch, and then data is added from the glue-patch to the corresponding nodes on block 2. On both source and target blocks are "O" and "X" reference points (solid circles and crosses in Fig. 9) which specify the location and orientation of the blocks with respect to the glue-patch. The reference points are used to perform the field component and indexing transformations between blocks.

Two types of field glue-patch are required:

- *surface patches* to exchange the two tangential field components at nodes common to the faces of two blocks;
- *line patches* to exchange the one tangential field component at nodes common to the edges of four or more blocks. If a node is common to M blocks, then it has associated with it $M(M-3)/2$ line patches.

Interior boundary conditions are applied by adding the glue-patch values d_{glue}^i to the corresponding node accumulator

$$d^i := d^i + d_{glue}^i \quad (101)$$

A simple 2-D example of the use of glue-patches is to apply doubly periodic boundary conditions to a rectangular domain. Four glue-patches are required: two surfaces patches to connect the north to south boundary and the east to west boundary, and two line patches to connect the NE to SW corner and the SE to NW corner. Glue-patch exchanges are required twice each timestep, once for d^i to complete the integration of Ampere's equation, and once for E_i to complete the

computation of E from d using the constitutive relationship (71).

Particle glue-patches. Particle exchange requires only surface patches. Each field surface glue-patch has a corresponding particle glue-patch. Particles passing through a glue-patch from a source block to a target block are passed from the storage areas of the source block particle tables to that of the target block via the glue-patch buffer. Position coordinates are transformed from the curved space components of the source block to those of the target block by using the locations of the "O" and "X" keys on the two connecting blocks (cf. Fig. 9). Particles passing through more than one block in a timestep are handled by repeated application of the surface glue-patch exchange.

4.2. Data input

4.2.1. Preprocessors

The generic preprocessor to treat general device geometries is called PEGGIE, for Parallel Electromagnetic Grid Generator Interface and Extensions. As implied by the word Interface, PEGGIE is not primarily intended to be a grid generator, although it has powerful in-built capabilities. The word Extensions relates to the ability to set most of the numerical, physical and housekeeping variables used by the PIC3D code.

The current version of PEGGIE accepts command line input. The use of a small number of tags, combined with FORTRAN 90 NAMELIST emulation makes it easy to specify complex objects in a compact way, set control and diagnostic parameters in a consistent manner, etc.

The in-built grid generator uses the concept that the device to be modelled is made by joining together simpler objects or blocks, each of which is cuboid in a suitably chosen, and in general non-orthogonal, coordinate system. Since element node positions are determined by transfinite interpolation, the point distribution along three adjacent edges defines the grid within the block. If the points are distributed uniformly in some (parametric) sense, then three integers n_i , $i = 1, 2, 3$ serve to define the local mesh. There are global constraints, arising from the need for adjacent objects to have the same point distribution on their common boundary, so the user has the ability to suggest n_i and let PEGGIE try to obtain a consistent set.

The algorithm employed is most easily explained for the case where the constituent blocks have entire faces in common (although partial joins are supported). Essentially a trail is started for each of the three adjacent edges that define a block. The connectivity or patch information on each surface implies a meshing in two coordinate directions for the adjacent block, i.e. two trails extend into it, and in general, another trail starts there. Successive blocks are examined until every trail has ended, typically by extending from one boundary to another. The trails are checked for consistency and where several suggested n_i lie upon a path, the one chosen is that corresponding to the earliest block in order of assembly.

It will be noted that blocks may have different orientations, e.g. the 1- and 2- coordinates in one block may correspond to respectively the 3- and 1- coordinates in an adjacent block. Such alignments are recorded by use of integer 3 vectors ($a\ b\ c$) corresponding to rotation matrices

$$\begin{pmatrix} i_a \\ i_b \\ i_c \end{pmatrix}$$

where i_a is a unit 3-vector with one entry, the a^{th} , non-zero and having the same sign as a , e.g. (1 -3 2) corresponds to the rotation matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}.$$

A similar technique has been devised independently [24]. The importance of the ($a\ b\ c$) notation is that quantities such as the n_i may be transformed simply by permutation and sign change.

Once blocks have been gridded, PEGGIE proceeds to generate first the covariant basis vectors e_i and then the coefficients $G_{ij}^{E,H}$ that describe the combination of block geometry and (in general) tensor permeability and permittivity as explained in Section 3.3.4. To reduce computing time, block symmetries are employed, so for example in an orthogonal cuboid only $G_{(i)(i)}$, $i = 1, 2, 3$ for one cell are actually calculated.

The G_{ij} serve to set a maximum timestep. Provided ϵ and μ are uniform, isotropic tensors, it may rigorously be shown for a grid consisting of identical parallelograms, that a necessary and sufficient criterion for stability is

$$c\Delta t \sqrt{\frac{g_{11} + g_{22}}{g}} \leq 1 \quad (102)$$

This contrasts with the criterion in Ref. [25] that has explicit off-diagonal terms, apparently because the 4-point sums involving off-diagonal G_{ij} have not been fully taken into account. (The off-diagonal terms appear implicitly in the volume element g .) In 3-D, for general but still homogeneous G_{ij} , it is necessary to resort to more heuristic arguments, which suggest that stability is achieved provided

$$c\Delta t \sqrt{\Gamma} < \frac{1}{\sqrt{3}} \quad (103)$$

where

$$\Gamma = \frac{1}{c^2} \max(G_{11}^E G_{22}^H, G_{22}^E G_{11}^H, G_{11}^E G_{33}^H, G_{33}^E G_{11}^H, G_{22}^E G_{33}^H, G_{33}^E G_{22}^H) \quad (104)$$

and this is used to specify Δt .

PEGGIE proceeds to calculate line patch information, needed where four or more blocks have a common edge. An important feature here is the representation of a block as the cube $[-1, 1]^3$ whose faces may be labelled by 3-vectors $(i_1, i_2, i_3) = (\pm 1, 0, 0), (0, \pm 1, 0)$ and $(0, 0, \pm 1)$ according to their surface normals. Edges can be represented by the 3-vectors that are the sums of two adjacent face vectors, and corners similarly as sums over the three faces that meet. These 3-vectors can each be condensed to one integer $i = 6i_3 + 2i_2 + i_1$ and the i values provide a natural ordering for respectively faces, edges and corners, e.g. DSWENU for faces where D, S, W, E, N and U represent the Down, South, West, East, North and Up faces of the cube $[-1, 1]^3$ (so D corresponds to the face with normal $(0, 0, -1)$, etc). A single integer operation decides which is the other cube face that shares a specified edge with a given face.

Answering the latter question is the basis of the line patch calculation: the need is, for each a block edge, to find a list of blocks which are coincident along at least part of that line, given only surface connectivity information. The idea is to describe a trail about the edge through the adjacent blocks, which may be a closed path, or may end at device boundaries, or even in general may split. The first stage of the algorithm is to rearrange the surface patch information, so in particular that for every edge of every block, there is

a list of contiguous surface patches. Starting with an unmarked edge a trail is pursued, marking each block edge (including the starting edge) that is found to be coincident with the first one, until it terminates as described above, or splits, in which case two separate trails are begun and pursued.

The last large set of non-local quantities to be computed is of the positions (x_{000}) of the blocks and their orientations (R) which respect to a global coordinate system. The position and orientation of one specified block defines this frame.

At this point PEGGIE calculates the scalings required to convert fields to the dimensionless units used by PIC3D. These are used at once in the generation of coefficients that treat boundary conditions of an applied electric field, resistive wall etc (see Section 3.3.6).

The timestep and scaled block dimensions are also employed to define control parameters; if the length of a PIC3D run is specified by the number of timesteps, PEGGIE works out suitable times for the saving of diagnostic information to disc. Line, surface and volume integral diagnostics (cf. Section 4.3) may be demanded, that may extend over different blocks.

Making full use of the compressed storage technique (Section 4.1.5), scalings are applied to e_i and G_{ij}^{EH} . Finally the remaining tables, such as the block to process mapping, (essential for PIC3D operations) are set, then the kernel data file suitable for immediate input to PIC3D is written to disc.

4.2.2. TWT preprocessor

The travelling wave tube (TWT) preprocessor is the first of a family of preprocessors which are designed for specialist application areas. It builds on existing experience of modelling TWTs in 2-D [26]. This preprocessor, which calls upon the library of routines developed for PEGGIE, provides a much simpler but more restricted method of initialisation.

The preprocessor was developed in collaboration with TWT tube engineers from EEV Ltd, and has been designed to take parameters with which the tube engineers prefer to work and generates the detailed input description that the main simulation code requires. The engineer is not required to set any parameters to define the multiblock and element subdivision of the tube interaction space, or the number of superparticles

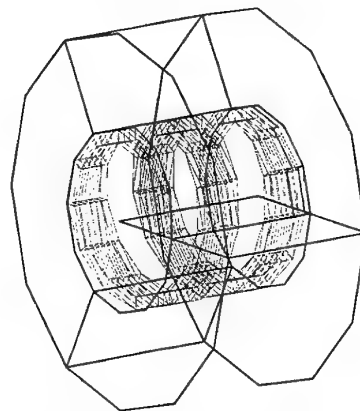


Fig. 10. A preprocessor output showing a section of the blocking of a TWT. Shown are a central cylindrical block containing three turns of the helix, surrounded by three further blocks. The thicker lines are block boundaries, and the thinner are helix element boundaries.

to be used, or any other numerical parameter to be used, although they can override the defaults if they wish.

The user input file (.uif) used by the TWT preprocessor uses the FORTRAN 90 NAMELIST format. An input file can include other files to allow users to set their own defaults, allowing a series of related calculations to be performed with the minimum amount of input. One important feature of the preprocessor is the ability to produce validation output to confirm that the main simulation code is correctly initialised; an example of one such output is shown in Fig. 10.

4.3. Data output

The main simulation code uses a hierarchical approach for specifying the output required; plot sets define the output file format, the field set, the domain set and the time set. The field set specifies a set of quantities to be plotted for each domain in the domain set, at the times listed in the time set. Each domain is a union of subdomains, each of which belong to a single block. The implementation is designed to allow new operations and output formats to be added easily.

The present version of the software has snapshot output for a limited range of geometries, and time-series output field specifications for point quantities, line integrals (e.g. $\int \mathbf{E} \cdot d\mathbf{l}$ or $\int \mathbf{H} \cdot d\mathbf{l}$), surface inte-

grals ($\int \mathbf{D} \cdot d\mathbf{S}$, $\int \mathbf{B} \cdot d\mathbf{S}$ or $\int \mathbf{j} \cdot d\mathbf{S}$) and volume integrals ($\int \psi dV$ where ψ may be for example particle number, $\mathbf{B} \cdot \mathbf{H}/2$, $\mathbf{E} \cdot \mathbf{D}/2$ or $\mathbf{j} \cdot \mathbf{E}$).

The implementation of the diagnostic output produces separate output for each process. Post processors can then combine data from several processes into a single dataset. This is satisfactory provided that the subdomains of a diagnostic domain set reside on the same process, but for more general situations message passing will also be needed for diagnostics.

5. Parallel benchmarking

The message passing model used in PIC3D matches the structure of massively parallel processors (MPPs) such as the Intel Paragon, IBM SP/2 and Meiko CS-2, and is also implemented efficiently on the parallel vector shared-memory computers such as the Cray Y-MP and C-90. PIC3D currently uses the Intel NX/2 library of communication subroutines (mostly CSEND and CRECV) because these are implemented with minimum overhead on many computers and are widely used. A version of the code using the popular PVM message passing language [27] has been written, and tested on an IBM SP/2. This gives maximum portability, but the scaling with increasing number of processors is unsatisfactory if Ethernet and the the Internet Protocol (IP) are used, due to their high message latency. However the PVM code is expected to scale satisfactorily when run under IBM's customized implementation PVMe [28].

Both 2-D and 3-D test cases have been used to evaluate parallel performance. The former, LPM2 benchmark uses the realistic device geometry of a magnetically insulated line oscillator (MILO), and the latter, LPM3 benchmark uses a 3-D periodic plasma to provide a clearly scalable test which does not depend on non-reproducible details such as random number generators.

5.1. LPM2 benchmarking

The choice of a 2-D MILO case for LPM2 was made because the geometry reflects realistic engineering calculation requirements and the results could be compared with those from other PIC codes [29,30]. Fig. 11 shows the electron scatter plot for a 20 block

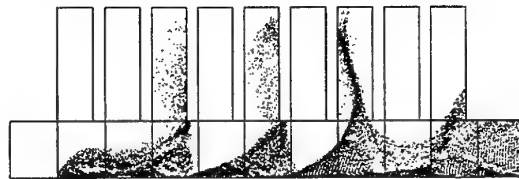


Fig. 11. An example of snapshot output of 2-D MILO simulation by PIC3D. This case is divided into 20 blocks: 9 cavities and 11 interaction space blocks.

subdivision of a MILO simulated using PIC3D in its 2-D mode.

LPM2 benchmarking runs undertaken on the Intel iPSC for a 64 block decomposition of a MILO, using a total of approximately 10,000 elements and 13,000 particles showed better than eighty percent of the theoretical speed-up for up to 16 processors, despite the non-uniform distribution of particles and small problem size. However, beyond this limit (i.e. < 4 blocks per process) the number of timesteps per second began to saturate (clearly, beyond 64 processors further speed-up is impossible without further problem subdivision!).

5.2. LPM3 benchmarking

The three-dimensional periodic plasma test case LPM3 (Local Particle Mesh benchmark # 3) was run on the Intel Paragon and Intel iPSC/860. This benchmark problem was chosen to be a representative test of the features of PIC3D on the available MPPs, to be easily scalable in size and to be physically realistic. LPM3 explores the capabilities of MPPs with more than a thousand processors.

In LPM3, the plasma space is divided into blocks, each of which consists of 512 particles representing the electrons, and 64 elements on which the fields are calculated. From the point of view of load balancing the parallel computer, the block is the smallest unit that can be allocated amongst the processors. The problem size is measured by the number of blocks N_b , and four problem sizes have been used with respectively $N_b = 8, 64, 512, 4096$. These correspond respectively to numbers of particles $N_p = 4K, 32K, 256K, 2M_2$ where $K = 1024$ and $M_2 = K^2$. The timestep is such that about ten percent of the particles transit to neighbouring blocks during a timestep. A run of 100

timesteps is chosen as the benchmark test because this takes only a few minutes for the problem sizes and number of processors of interest, and the conservation of total number of particles is used as a validity check.

There are two different versions of the benchmark code, which are selected by an input variable. The *per-patch* version sends a separate message for every patch in the system, and there are 9 patches for every block. In the *per-process* version, on the other hand, the patch messages are assembled and sorted in a buffer so that only one message is sent to every other process to which a given process is attached. The per-process code may send 10 or 100 times fewer messages than the per-patch version, and so should be significantly faster than the per-patch version on computers with a high message start-up time or latency. This difference is clearly seen when the code is run on the IBM SP/2 using the high latency PVM-IP message passing interface. For computers with low latency such as the Intel Paragon using NX/2 under SUNMOS, there is little difference between the versions.

Results: All the benchmark measurements obtained are shown in Fig. 12. The results are expressed as temporal performance R_T in units of timestep per second ($tstep\ s^{-1}$), an unambiguous metric which expresses exactly what a program user needs to maximise [31]. We have deliberately not used the sometimes popular metrics of Parallel Speed-up and Efficiency because these are not absolute measures and cannot be used correctly to compare the performance of different computers [31–33]. Neither have we expressed the results as Megaflop per second, because this would make the curves lie closely on top of each other, and disguise the actual time of computation from the reader. In our units of timestep per second, calculations which take the same time lie at the same height in the graph; faster calculations lie higher and slower lie lower. These statements would not necessarily be true if the results were expressed as speed-up or in megaflop per second.

For each block size there are two pairs of curves. The pair closest to the theoretical dotted line is that for the Intel Paragon running under the Sandia SUNMOS 1.4.8 operating system, and the pair about 60 percent lower and slower are for the iPSC/860 running under the NX release 3.3.2 operating system. For each computer the open symbols are the measured val-

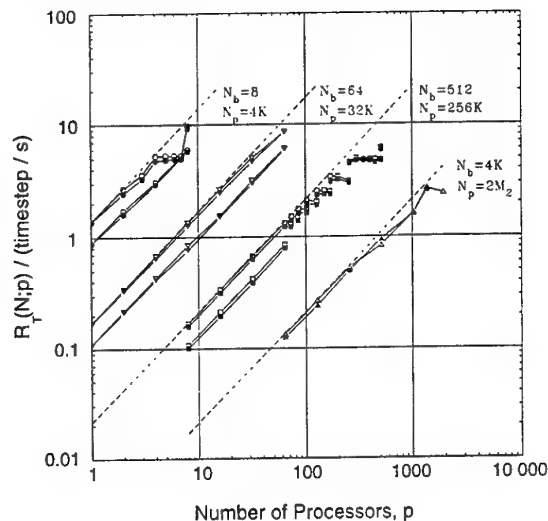


Fig. 12. Temporal Performance of the LPM3 benchmark measured in units of timestep per second, for four problem sizes and up to 1840 processors on the Sandia Laboratory's Intel Paragon XP/S 140 (upper pair of curves which are close to the dotted lines), and up to 64 processors on the Intel iPSC/860 (lower pair of curves). See text for further details.

ues using the per-process code and the corresponding filled symbols for the per-patch code. The fact that the curves for both versions are almost the same on both the computers, means that message latency is not a problem with the Paragon under SUNMOS or with the iPSC/860. The latency has separately been measured using the COMMS1 'pingpong' benchmark [33] to be about $80\ \mu s$ for both these computers.

The dotted lines are the theoretically perfect linear scaling predictions for the Paragon, calculated from the one-processor performance on the eight block problem. The theoretical scaling assumes that performance is proportional to the number of processors, and inversely proportional to the problem size; ideally, p -times as many processors should compute an existing problem p -times faster (the speed-up), or a problem p -times bigger in the same time (the size-up). Alternatively, of course, the increase in number of processors should be able to be used to obtain a combination of speed-up and size-up. The stepwise nature of the measured performance curves arises due to load imbalance when the number of blocks is not exactly divisible by the number of processors. If we

concentrate attention on the best performance figures which correspond to perfect load balance, when every processor is computing the same number of blocks, we find that the measured performance is within 80 percent of the ideal linear scaling except for the 512 block case with greater than 256 processors, and for the 4096 block case with more than 1366 processors. For these cases with larger numbers of processors, performance saturation is beginning to be seen.

For the set of cases shown in Fig. 12 it is evident that there is not enough parallelism in the problem to obtain more than about 10 tstep s^{-1} . The 8 block problem can at most use 8 processors, and there is no possibility of using more processors to speed up that problem. The best we can do with the extra processors is to solve bigger problems in about the same time, and this is seen for the cases of $N_b = 64, 512, 4096$, none of which exceed 10 tstep s^{-1} however many processors are used.

6. Final remarks

We have devised and implemented new algorithms for electromagnetic Particle-in-Cell (PIC) calculations to enable the simulation of realistic devices with complex geometry. These algorithms build on existing experience with the electromagnetic PIC approach; indeed in simple geometries they reduce to state-of-the-art algorithms. They are economical in that under such circumstances they cost no more, and in that they perform efficiently on virtually all types of computer, i.e. on single and multiple processor machines which may or may not share memory. Moreover many features of the algorithms serve to reduce software costs, by making it easy to change the properties of a model by, say, altering its dielectric properties or surface material.

More precisely, our algorithms extend the finite element method of Lewis [16] to more general element shapes and to the time variable. These methods avoid the quadrature problems encountered in earlier generalisations [34], whilst automatically ensuring charge conservation. The variational formulation gives algorithms with the “energy conserving” rather than the “momentum conserving” property. If desired, similar momentum conserving schemes can be obtained in general geometry without losing the charge conserva-

tion property by introducing a field reconstruction step before interpolating force to particle positions.

We have focused in this paper on elements which transform to unit cubes in some general coordinate system; these were chosen for reasons of elegance and computational efficiency. However elements of higher order and of different shape could be treated. With cuboid elements, the tensor formulation provides simple expressions, differing little from those obtained using finite differences on a Cartesian mesh, for the update of fields and particle motion, yet allows fully body-fitted subdivision of the computational domain. The problems of locating the element in which a particle lies, of computing its acceleration and associated current, and of integrating Maxwell’s equations, are no more difficult than in Cartesian PIC.

The form of the numerical algorithm allows data to be organised straightforwardly into a form suitable for distributed-memory parallel computers. This has made it possible for 3DPIC to be coded in FORTRAN 77, rather than have to submit to the stricter programming regime of C++ and endure its shortcomings regarding particularly speed of execution [35].

The physics of Maxwell’s equations provides a natural organisation of the data, in that interactions at a point in space only involve information from its light cone in space-time. Our software uses this fact to divide a large PIC computations into many smaller ones which communicate only with their immediate neighbour. Hence work can be shared evenly amongst the processors of a MIMD machine, whilst minimising the amount of global data and interprocessor message passing. In addition, the simple logical cube addressing within each block leads to fast serial processing within each slave process.

Benchmarking confirms that our multiblock data organisation offers large computational intensity and weak Amdahl limit on parallel processor speed-up [14]. In particular, we have shown that doubling the size of the computer allows simulations of twice the size to be undertaken in the same elapse time. A similar approach [36] to dividing the computational domain has demonstrated very high efficiencies for PIC calculations in simple geometries.

An alternative to the hierarchical multiblock approach is to use a completely unstructured mesh [37–41]. This has the advantage that topologically difficult objects can be more readily meshed, but the disadvan-

tages that the large data compressions and the ease with which particles can be tracked in our multiblock method are not readily attainable.

The preprocessor, simulation and postprocessor software packages described in this paper are not yet fully mature, but are sufficiently well advanced to show that they will provide effective tools for computer simulation studies of not only parts of microwave devices, but given suitable MPPs, of entire microwave systems with hitherto inaccessible scale lengths.

Acknowledgements

Research sponsored in part by Air Force Office of Scientific Research (AFSC) under contract F61708-93-C-001, by AEA Corporate Research Funding and through subcontract by EEV Ltd under DRA Malvern contract number A 376 415677 E5W. Some of the work reported was undertaken under earlier AFSC contract F49620-92-C-0035 and under contract RAE 1B/7 sponsored by DRA Farnborough.

References

- [1] R.W. Hockney and J.W. Eastwood, *Computer Simulation using Particles* (Adam-Hilger, Bristol, 1988).
- [2] C.K. Birdsall and A.B. Langdon, *Plasma Physics via Computer Simulation* (Adam-Hilger, Bristol, 1991).
- [3] D.W. Hewett, *Comput. Phys. Commun.* 84 (1994) 243–277.
- [4] J.W. Eastwood, in: *Proc. 7th Ann. Rev. Prog. App. Computational Electromagnetics* (Naval Postgraduate School, Monterey, March 1991) pp. 655–660.
- [5] J.W. Eastwood, *Computer modelling of microwave sources*, in: *Proc. 18th Annual Conf. Plasma Phys.* (IoP, Colchester, 1991) pp. 35–42.
- [6] W. Arter and J.W. Eastwood, *Electromagnetic modelling in arbitrary geometries by the virtual particle-mesh method*, in: *Proc. 14th Int. Conf. Num. Sim. Plasmas* (APS, Annapolis, September 1991) Paper PWE15.
- [7] J.P. Boris, *Relativistic plasma simulation – optimization of a hybrid code*, in: *Proc. 4th Conf. Num. Sim. Plasmas* (NRL, WA, 1971) p. 3.
- [8] B. Goplen, L. Ludeking, J. MacDonald, G. Warren and R. Worl, *MAGIC Reference Manual–Algorithms*, MRC/WDC-R-201 (Mission Research Corporation, October 1989).
- [9] B. Goplen, L. Ludeking, D. Smithe and G. Warren, *User-configurable MAGIC for electromagnetic PIC calculations*, *Comput. Phys. Commun.* (1995), this issue.
- [10] B. Goplen, J. MacDonald and G. Warren, *SOS Reference Manual*, MRC/WDC-R-190 (Mission Research Corporation, March 1989).
- [11] A.T. Drobot, C.-L. Chang, K. Ko, A. Mankofsky, A. Mondelli, L. Seftor and P. Vitello, *IEEE Trans. Nucl. Sci.* NS-32 (1985) pp. 2733–2737.
- [12] D.B. Seidel, M.L. Kiefer, R.S. Coats, T.D. Pointon, J.P. Quin-tenz and W.A. Johnson, *Multitasking the 3-D electromagnetic PIC code QUICKSILVER*, in: *Proc. 13th Conf. Num. Sim. Plasmas* (Santa Fe, NM, September 1989) Paper IM6.
- [13] K.S. Yee, *IEEE Trans. Antennas Propagat.* 14 (1966) 302–307.
- [14] R.W. Hockney and C.R. Jesshope, *Parallel Computers 2: Architecture, Programming and Algorithms* (Adam Hilger, Bristol, 1988).
- [15] J.W. Eastwood, *Comput. Phys. Commun.* 64 (1991) 252.
- [16] H.R. Lewis, *Methods Comput. Phys.* 9 (1970) 307.
- [17] H. Goldstein, *Classical Mechanics* (Addison-Wesley, Reading, MA, 1950).
- [18] W. Arter and J.W. Eastwood, in: *Mathematical and Numerical Aspects of Wave Propagation Phenomena*, eds. G. Cohen, L. Halpern and P. Joly (SIAM, Philadelphia, 1991) p. 719.
- [19] M.R. Spiegel, *Vector Analysis* (Schaum, New York, 1959).
- [20] W.J. Gordon and C.A. Hall, *Int. J. Numer. Methods Eng.* 7 (1973) 461.
- [21] A.D. Burns, N.S. Wilkes, I.P. Jones and J.R. Kightley, *FLOW3D: body-fitted coordinates*, AERE-R12262 (AEA Harwell, UK, 1986).
- [22] A.D. Burns and N.S. Wilkes, *A finite difference method for the computation of fluid flows in complex three-dimensional geometries*, AERE-R12342 (AEA Harwell, UK, 1986).
- [23] B.B. Godfrey, *J. Comput. Phys.* 15 (1974) 504–521.
- [24] A. Rizzi, P. Eliasson, I. Lindblad, C. Hirsch, C. Lacor and J. Haeuser, *Comput. Fluids* 22 (1993) 341.
- [25] J.-F. Lee, R. Palandech and R. Mittra, *IEEE Trans. Microwave Theory Tech.* 40 (1992) 346.
- [26] N.J. Brealey, in: *High Power Microwave Generation and Applications*, eds. E. Sindoni and C. Wharton (SIF, Bologna, 1992) p. 549.
- [27] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manckek and V. Sunderam, *PVM 3.0 User's Guide and Reference Guide*, ORNL report (Oak Ridge National Laboratory, Oak Ridge, TN, 1993).
- [28] R.W. Hockney, *The communication challenge for MPP: Intel Paragon and Meiko CS-2*, *Parallel Computing* 20 (1994) 389–398.
- [29] J.W. Eastwood, *Computer modelling of high power microwave sources*, in: *High Power Microwave Generation and Applications*, eds. E. Sindoni and C. Wharton (SIF, Bologna, 1992) pp. 539–548.
- [30] M. Amman, private communication (1993).
- [31] R.W. Hockney, *A framework for benchmark performance analysis*, *Supercomputer 48 IX* (1992) 9–22. (Also published in: J.J. Dongarra and W. Gentzsch, eds. *Computer Benchmarks* (Elsevier, Amsterdam, 1993) pp. 65–76.)
- [32] D.H. Bailey, *Twelve ways to fool the masses in performance evaluation*, *Supercomputer 45 VII* (1991) 4–7.

- [33] R.W. Hockney and M. Berry, eds, PARKBENCH report: public international benchmark for parallel computers, Sci. Program. 3 (1994) 101–146.
- [34] B.B. Godfrey, Application of Galerkin's method to particle in cell plasma simulation, in: Proc. 8th Conf. Num. Sim. Plasmas (Monterey, CA, 1978).
- [35] D.W. Forslund, C. Wingate, P. Ford, J.S. Junkins, J. Jackson and S.C. Pope, in: Proc USENIX C++ Conference (USENIX Associates, Berkeley, 1990) p. 177.
- [36] P.C. Liewer and V.K. Decyk, J. Comput. Phys. 85 (1989) 302.
- [37] D. Seldner and T. Westermann, J. Comput. Phys. 79 (1988) 1.
- [38] R. Lohner and J. Ambrosiano, J. Comput. Phys. 91 (1990) 22.
- [39] F. Assous, P. Degond and J. Segre, Comput. Phys. Commun. 72 (1992) 105.
- [40] F. Hermeline, J. Comput. Phys. 106 (1993) 1.
- [41] F. Assous, P. Degond, E. Heintze, P.-A. Raviart and J. Segre, J. Comput. Phys. 109 (1993) 222.

G MAUI-3DPIC/README-MIMD

05/09/15
15:48:21

readme-mimd

1

File: ~roger/MAUI-3DPIC/README-MIMD

MIMD VERSION PIC3D

OPERATING INSTRUCTIONS

MAUI HPCC

Roger W. Hockney
24 August 1995

1) PVM3 PARALLEL VERSION

The MIMD version of PIC3D has initially been set up under public domain PVM3. It is recognised that this is not the most efficient implementation of message passing or of PVM, but PVM3 is widely available and this version should be easily portable. Experience on the Maui facility is that PVM3 is the most robust message passing system and does not need dedicated nodes. It is thus the easiest to use for test purposes. Message passing interfaces with significantly less message latency are IBM's native message passing library (MPL) and IBM's efficient implementation of PVM (called PVMe). Conversion to MPL and PVMe is considered a second stage and should not be too difficult, however both systems need dedicated nodes on the SP2 which can be difficult to get. Testing of these versions will therefore be slower than PVM3.

2) LOGING IN TO MAUI

In order to login to Maui from Culham, do the following:

% is the Unix prompt at both Culham and Maui,
<PWD> is the password, and
. . . means some lines are omitted

rfsunc% xterm& Start an 'xterm' window, otherwise you will
 not be able to edit with 'vi' properly.

I usually open three xterm windows, one for editing and recompilation, and two following the Unix processes on two of the nodes forming the PVM computer using the 'ps' command. The next login procedure must be done in each window. The node allocated to your login is chosen by the system, but afterwards you can 'rlogin' to another node.

The full name for each node (there are 400 of them) is of the form

fr<frame number>n<node number>.mhpcc.edu

Thus 'fr8n09.mhpcc.edu' is node 9 of frame 8
This full name is usually necessary if it is requested.
A frame either holds 16 'thin' nodes numbered 1,2,3,...,16, or
8 'fat' nodes numbered 1,3,5,...,15.

The full Internet name of the Maui SP2 is:

tsunami.sp2.mhpcc.edu (164.122.163.114)

A 'tsunami' is evidently one of those huge tidal waves that hit the islands from time to time. Maui HPCC was shut down for a day last year because of a real one of these being forecast.

This is the login procedure:

rfsunc% telnet gw In the xterm window, telnet through the
 Harwell gateway computer

. . .
gw telnet proxy (Version V1.3) ready:
Username: rhockney
Password: <PWD>
Login Accepted
tn-gw-> c tsunami.sp2.mhpcc.edu
Trying 164.122.163.114 port 23...

telnet (fr?n??.mhpcc.edu) The system tells you which frame
 and node number has accepted your
 login.

05/09/15
15:48:21

readme-mimd

2

```
. . .
AIX Version 3
(C) Copyrights . . .
login: hockney
hockney's Password: <PWD>
```

```
. . .
Last login: . . .
```

```
% ls                               Unix prompt from Maui SP2, then 'ls'
                                   The following directories concern us:

    MAUI-3DPIC                     MIMD PVM3 version of PIC3D, subdirectories
                                   below this copy those at Culham.

    SP2                           LPM3 benchmark code used for testing
                                   message-passing interface.

    pvm3/bin/RS6K                 Directory for binary executables, expected
                                   by PVM3. The pic3d executable MUST be
                                   copied to here after any recompilation.

% hostname                         Type at any time to find out your node's
fr8n08.mhpcc.edu                  name.
```

3) SETTING UP PVM3

It is first necessary to set up a PVM3 parallel machine (here called your PVM) from the available nodes of the IBM SP2. Perform the following steps on the node of the SP2 that has accepted your login, here assumed to be node fr8n08.mhpcc.edu. For example to get a parallel machine with 2 nodes:

```
% pvm                             Start PVM3 daemon on your login node, and also
                                   start the PVM console on the same node.
                                   The console is used to control the PVM,
                                   add and remove nodes, test status of the PVM,
                                   etc. The console's prompt is 'pvm>', and the
                                   available commands are given by typing 'help'.
```

```
pvm> conf                         List the configuration of your PVM.
```

```
1 hosts, 1 data format
      HOST      DTID      ARCH      SPEED
fr8n08.mhpcc.edu 40000    RS6K      1000
```

It comprises just the one node that we logged on to.
Now we add 1 more node:

```
pvm> add fr8n09.mhpcc.edu
1 successful
      HOST      DTID
fr8n09.mhpcc.edu 100000
```

This starts a PVM3 daemon on the second node and thereby adds it to your PVM configuration. To confirm that we have a PVM configuration of two nodes, type 'conf':

```
pvm> conf                         List the configuration of your PVM.
```

```
2 hosts, 1 data format
      HOST      DTID      ARCH      SPEED
fr8n08.mhpcc.edu 40000    RS6K      1000
fr8n09.mhpcc.edu 100000    RS6K      1000
```

```
pvm> quit                         Quit the PVM console,
pvmd still running                but leave PVM running, i.e. your PVM computer
                                   is there but you just do not have its console
                                   running. The console can be restarted at any time
                                   by typing 'pvm' to the Unix prompt of any node
                                   that is part of the PVM.
```

Your PVM is now setup, and a parallel code can be started by executing one copy on either of the two nodes of the PVM. The copy started will detect that it is the first or 'master' copy and whether it needs to spawn off other copies of the code on other nodes to make up the number

05/09/15
15:48:21

3

readme-mimd

of processes NPRES specified in the data file. If NPRES > 1 these extra processes are spawned and started. The parallel code is then running. Remember that PVM takes the executable to be used in the spawned copies from the directory ~/pvm3/bin/RS6K, so after a recompilation the executable must be copied to this directory if you want it to be used by PVM! There can be much confusion if you forget to do this.

The processes are assigned to the nodes of your PVM in the sequence given by the 'conf' command, so that if the total number of processes equals the number of nodes of the PVM then there will be one process per node, as desired. If there are more processes than nodes, the nodes are reused in a cyclic fashion, and there will be more than one PIC3D process executing on each PVM node. This is OK because each PIC3D process is implemented as a separate Unix process which can be seen if you login to a node and do a 'ps', e.g:

```
% ps -elf | grep hockney
```

In reply, the PVM daemon that runs on every node of the PVM and holds the PVM together, shows up as:

```
. . . /source/pd/pvm3.2.6/pvm3/lib/RS6K/p
```

and, if running, the console shows up as an additional process

```
. . . pvm
```

The PIC3D process shows up as

```
. . . . . pic3d
```

In fact, if you execute the PVM code with NPRES=128 on a PVM of one node you will find 128 copies of the executable 'pic3d' running on the node.

If the PVM console fails to add nodes when requested, or fails to start, do a 'ps' and remove any old PVM daemons that might have been left from previous sessions that were not ended cleanly. Also go to /tmp and remove any files belonging to 'hockney' and called:

```
/tmp/pvmd.544  
/tmp/pvml.544
```

Don't ask, just do it!

To end a PVM session cleanly, invoke the PVM console, and type 'halt':

```
% pvm  
pvmd already running.  
pvm> halt  
libpvm [t40002]: mxfer() EOF on pvmd sock
```

That should avoid many problems when you try to setup a PVM next time.

When you setup a PVM computer in the above way, it is well to remember that your PVM may overlap many other users PVMs, and when you do a 'ps -elf' you may see many other PVM daemons belonging to other users. This allows many parallel programs to be tested by many users without requiring a set of dedicated nodes to be assigned to each user. It also means however that your PVM is time-sharing the CPU of each node with other users, and therefore no meaning can be attached to the execution times that are recorded. The only way to do proper timing is to make application for the use of nodes dedicated to your application alone.

All the above is a bit tedious, but is how public-domain PVM3 is used. Many computer manufacturers' versions of PVM3 are simpler, and therefore probably better, in that they do not allow the PVM3 spawning instruction. Their own operating system takes over the responsibility of providing the required number of nodes, and the users executable is started on each of these nodes. The Cray T3D and Meiko CS2 implementations are like that. However here we have described the portable public-domain PVM3 from Oakridge as is available free by anonymous ftp or from the Web, and commonly used on workstation clusters.

4) DIRECTORY STRUCTURE AND TEST CASES

4.1 The Executable

05/09/15
15:48:21

readme-mimd

4

The executable, 'pic3d', for the parallel version of PIC3D is in directory:

~hockney/MAUI-3DPIC/PIC3D

and copied for use by the PVM3 spawn subroutine into:

~hockney/pvm3/bin/RS6K

4.2 Test Data and Results

The data and results for various test cases are in directories, which are repeated also in the Culham directories under ~roger. For details of the test cases see the report:

AEA/TYKB/28006/TN/2: "Demonstration Calculations using the ElectroMagnetic PIC Software Suite 3DPIC"
by Eastwood, Arter, Brealey and Hockney, September 1995.

Each test case has a directory (ref) for the single processor reference results produced with the single processor code on a SUN workstation at Culham, and three directories (1-proc, 2-proc and 8-proc) for results from the MIMD parallel PIC3D code produced at Maui. These are for the MIMD code running respectively on 1, 2 and 8 processors (or nodes). The maximum number of processors that can be used is equal to the number of blocks, so this is also given below:

4.3) Wave Test Case, une51, with 10 timesteps per period (8 blocks), ----- for 100 steps:

~hockney/MAUI-3DPIC/PIC3D/UNE51/Datum/ref
 /1-proc
 /2-proc
 /8-proc

4.4) Wave Test Case, une51, with 100 timesteps per period for 1000 steps. ----- This was setup to see the effect of sampling errors on the wave:

~hockney/MAUI-3DPIC/PIC3D/UNE51/SmallDT/ref
 /1-proc
 /2-proc
 /8-proc

4.5) Single Particle Orbit Test, porb21 (8 blocks) for 400 steps: -----

~hockney/MAUI-3DPIC/PIC3D/PORB21/Datum/ref
 /1-proc
 /2-proc
 /8-proc

4.6) Full MILO Simulation (72 blocks) for 2000 steps: -----

~hockney/MAUI-3DPIC/PIC3D/MLS407/Datum/ref
 /1-proc
 /2-proc
 /8-proc

5) RUNNING MIMD PIC3D

After starting PVM as described above, PIC3D can be run in various ways, but I have adopted the following procedure. Starting from the ~hockney top directory:

% rm une51*	Remove leftover files
% rm fort*	from the top directory.
% cd MAUI*/PIC3D/UNE51/Datum/2-proc	Go to test directory
% cp une51.dat.p2.datumDT input.dat	Copy input data to input.dat
	The current MIMD version for various reasons has the input file wired in as 'input.dat'.

95/09/15
15:48:21

5

readme-mimd

This can be changed later.

% rm une51*

Remove previous results.
Failure to remove old *.tsd
files will stop the program
from running.
BUT LEAVE any *.dat* for
future use.

% rm fort*

% pic3d

PIC3D should run taking its
data from 'input.dat'

Some test printouts are still in the program, but the output should look
something like the following, with a write to the screen every 10 steps
to show progress. The pvmfjoin error does not seem to matter:

```
MAIN.F started
Enter MCDPVM3/RHSPAW: K= -1
Befor NODASG(0) MYPRES = 1  NPRES= 2
Enter MCDPVM3/RHSPAW: K= 0
After pvmfspawn: ntids(1)= 262151  inumt= 1
libpvm [t40006]: gs_getgstdid() failed to start group server: No such file
ERROR in pvmfjoin, RHBARR, inum = -14  MYPRES = 1
After NODASG(0) MYPRES = 1  NPRES= 2
```

```
-----
Parallel 3DPIC
PVM3 Version
August, 1995
-----
```

```
3D E/M PIC Simulation
Eastwood, Arter, Brealey and Hockney
AEA Technology, Culham Laboratory
Comput. Phys. Comm. 87, 155-178 (1995)
-----
```

```
Number of Processors = 2
Active MYPRES = 1  NPRES= 2
CNOOUT =une51.outp2:1
NLRES = F  MYPRES = 1
. . .
. . .
. . .
. . .
```

MYPRES = 1

```
Beg Step= 70 mypres= 1 npres= 2
Beg Step= 80 mypres= 1 npres= 2
Beg Step= 90 mypres= 1 npres= 2
Beg Step= 100 mypres= 1 npres= 2
```

```
Elapsed time this run = 2.680000 sec
                        or = .044667 min
                        or = .000744 hour
```

%

Output files from processor with MYPRES=1 will appear in the present
test directory, but output from all other processors (MYPRES=2,...,NPRES)
will be found in the top directory. To put all output files in the test
directory do a move:

```
% mv ~/une51* .      Allow une51.grd from top directory to
                       overwrite that in test directory. It
                       contains nothing of interest. We should
                       give the *.grd files different names from
                       each processor, to stop this overwrite.
```

6) RETURNING RESULTS TO CULHAM

The graphical output files can best be examined at Culham using the
MPICTIM and ghost graphics. The above file structure is repeated at
Culham under ~roger instead of ~hockney, so the procedure is to copy
all files under the Maui test directory to the same directory at Culham.
This is done using 'ftp' from Culham through the Harwell gateway as
follows:

rfsunc% cd

95/09/15
15:48:21

6

readme-mimnd

```
rfesunc% cd MAUI*/PIC3D/UNE51/Datum/2-proc      Go to test directory at Culham
rfesunc% ftp gw                                  FTP through Harwell gateway
Connected to gw.aea.orgn.uk.
220-Proxy first requires authentication
220 gw FTP proxy (Version V1.3) ready.
Name (gw:roger): rhockney
331 Enter authentication password for rhockney
Password: <PWD>
230 User authenticated to proxy
ftp> user hockney@tsunami.sp2.mhpcc.edu
331-(-GATEWAY CONNECTED TO tsunami.sp2.mhpcc.edu---)
331-(220 fr3n02.mhpcc.edu FTP server (Version 4.14 Fri Aug 5 13:39:22 CDT 1994) ready.)
331 Password required for hockney.
Password: <PWD>
230 User hockney logged in.
ftp> binary                                     Don't forget BINARY mode for
                                                graphics files

200 Type set to I.
ftp> hash                                       Hash to follow progress
Hash mark printing on (8192 bytes/hash mark).
ftp> cd MAUI-3DPIC/PIC3D/UNE51/Datum/2-proc
250 CWD command successful.
ftp> mget *
mget input.dat? y
200 PORT command successful.
150 Opening data connection for input.dat (97210 bytes).
#####
226 Transfer complete.
local: input.dat remote: input.dat
97210 bytes received in 71 seconds (1.3 Kbytes/s)
mget une51-01-0001.tsd? y
200 PORT command successful.
150 Opening data connection for une51-01-0001.tsd (4512 bytes).
#
226 Transfer complete.
local: une51-01-0001.tsd remote: une51-01-0001.tsd
4512 bytes received in 4.6 seconds (0.96 Kbytes/s)
mget une51-01-0002.tsd? y
. . .
mget une51-02-0001.tsd? y
. . .
mget une51-02-0002.tsd? y
. . .
mget une51-03-0001.tsd? y
. . .
mget une51-03-0002.tsd? y
. . .
mget une51.dat.p2.datumDT? n
mget une51.grd? y
. . .
mget une51.outp2:1? y
. . .
mget une51.outp2:2? y
. . .
ftp> quit
221 Goodbye.
rfesunc%
```

7) RECOMPILATION

After any changes to the source, the program should be recompiled to make a new executable, pic3d, in the usual way. This should be done in the PIC3D directory, as follows. The example shows a recompile of auxval.f and gstplt.f followed by relinking:

```
% cd MAUI*/PIC3D
% make
f77 -O -qcharlen=2000 -c auxval.f                Recompile
** auxval === End of Compilation 1 ===
1501-510 Compilation successful for file auxval.f.
      cd ../DIAG; make libdiag.a;
f77 -O -qcharlen=2000 -c gstplt.f                Recompile
** gstplt === End of Compilation 1 ===
1501-510 Compilation successful for file gstplt.f.
      ar r libdiag.a gstplt.o
      touch libdiag.a
```

5/19/15
5:18:21

readme-mimd

7

```
cd ../NETGLB; make libnetglb.a;
Target libnetglb.a is up to date.
cd ../NETBLK; make libnetblk.a;
Target libnetblk.a is up to date.
cd ../PARGLB; make libparglb.a;
Target libparglb.a is up to date.
cd ../PARBLK; make libparblk.a;
Target libparblk.a is up to date.
cd ../EMGLB; make libemglb.a;
Target libemglb.a is up to date.
cd ../EMBLK; make libemblk.a;
Target libemblk.a is up to date.
cd ../PEGLIB; make libpeglib.a;
Target libpeglib.a is up to date.
cd ../GLBLIB; make libglblib.a;
Target libglblib.a is up to date.
cd ../MCDPVM3; make libmcdlib.a;
Target libmcdlib.a is up to date.
cd ../CRONUS; make libcronus.a;
Target libcronus.a is up to date.
cd ../OLYPAR; make libolypar.a;
Target libolypar.a is up to date.
cd ../OLYMPUS; make libolympus.a;
Target libolympus.a is up to date.
f77 -O -qcharlen=2000 -o pic3d main.o advb.o auxval.o bainit.o binit.o btoh.o bufunl.o ccinit.o
filtr.o cotrol.o ctdepc.o curode.o ddot.o ddot2.o dfiltr.o dtoe.o ebext.o einit.o embed0.o embed2.o
ndrun.o expt.o helmh.o inital.o labrun.o mindat.o move1.o move2.o newp.o output.o painit.o restor.o
estrtr.o reth.o stepon.o strage.o tesend.o -L../DIAG -L../NETGLB -L../NETBLK -L../PARGLB -L../PARBLK -L.
/EMGLB -L../EMBLK -L../PEGLIB -L../GLBLIB -L../MCDPVM3 -L../CRONUS -L../OLYMPUS -L../ghost/lib -ldi
g -lnetglb -lnetblk -lparglb -lparblk -lemglb -lemglb -lemblk -lpeglib -lgblib -lmcdlib -l
ronus -L../OLYPAR -lolypar -lolympus -lghost -lgrid -L/source/pd/pvm3.2.6/pvm3/lib/RS6K -lfpvm3 -lpvm3 -lgpv
3
Relink
```

```
% mv pic3d ~/pvm3/bin/RS6K
overwrite /u/hockney/pvm3/bin/RS6K/pic3d? y
Move new executable to correct
PVM directory, for use by
pvmfspawn.
```

%

END README-MIMD FILE

H README-LPM3

05/09/12
4:10:28

README-LPM3

1

File: ~/hockney/SP2/LPM3/test/README-LPM3

LPM3 Benchmark

Roger Hockney
September 1995

(1) RUNNING LPM3

To run the LPM3 Benchmark, type at the UNIX prompt (%) on a node of the Maui SP2:

```
% cd ~/hockney/pvm3/bin/RS6K      Go to pvm3 executable directory
% cp lpm3.pvm3.3Aug95 lpm3        Copy executable to name expected by
                                  lpm3 code
% cd ~/hockney/SP2/LPM3/test      Go to directory to be used for test
% cp neut50.dat input.dat        Copy data file to working data file.
                                  We do the 8-block test as an example.
                                  Other data files are:
```

```
neut50.dat    8-block test
neut51.dat    64-block test
neut52.dat    512-block test
neut53.dat    4096-block test
```

```
% vi input.dat                    Edit 2nd line to number of
                                  processors as required, up to the a
                                  maximum equal to the number of blocks.
```

```
% pvm                            Start a PVM machine and add as many
                                  processors as required.
```

```
pvm> add . . .
pvm> add . . .
pvm> add . . .
```

```
pvm> quit                        Quit PVM Console
```

```
% lpm3                          Start lpm3
```

All the important output goes to the screen with one line printed every 10 steps. The timing results are given at the end in the

```
*****
LPM3 - RESULTS SUMMARY
*****
```

The key benchmark numbers are, for example:

Number processors, NPRES = 2

Elapsed Benchmark wall time = 55.62006760 s

Temporal Performance = 1.797912240 tstep/s

This output is repeated at the end of the output from the first processor in file:

o_neut50p2:1

The second processor's output file appears in the top directory

```
% mv ~/o_* .                    Move output from other processors
                                  to test directory
```

(2) SCREEN OUTPUT

The screen output for the case neut50.dat (8 blocks) is in file

lpm3.out

and follows. The initial ERROR does not seem to matter:

```
ERROR in pvmfjoin, RHBARR, inum = -14 MYPRES = 1
```

95/09/12
14:10:28

README-LPM3

2

LPM3 BENCHMARK

3D Periodic PIC Plasma
Roger Hockney, July 1994

Number of Processors = 2

NSTEP= 0 MYPRES= 1 : *** PBLOCK, c1s6b.f: MAXBLK OK ***
IBLMAX= 8 .LE. MAXBLK= 5096
NSTEP= 0 MYPRES= 1 : *** PBLOCK, c1s6b.f: MAXPCH OK ***
IPAMAX= 72 .LE. MAXPCH 40000

Random Number test: first 20

NSEED= 0 NSW= 0
RANDF(0)= 0.1809234619 NSEED= 11857
RANDF(0)= 0.3858184814 NSEED= 25285
RANDF(0)= 0.6827545166 NSEED= 44745
RANDF(0)= 0.6078643799 NSEED= 39837
RANDF(0)= 0.5761871338 NSEED= 37761
RANDF(0)= 0.5847930908 NSEED= 38325
RANDF(0)= 0.4784088135 NSEED= 31353
RANDF(0)= 0.2754211426E-01 NSEED= 1805
RANDF(0)= 0.6910705566E-01 NSEED= 4529
RANDF(0)= 0.9595489502 NSEED= 62885
RANDF(0)= 0.5904693604 NSEED= 38697
RANDF(0)= 0.2167510986 NSEED= 14205
RANDF(0)= 0.3471832275 NSEED= 22753
RANDF(0)= 0.9475860596 NSEED= 62101
RANDF(0)= 0.2064361572 NSEED= 13529
RANDF(0)= 0.1129913330 NSEED= 7405
RANDF(0)= 0.9791564941E-01 NSEED= 6417
RANDF(0)= 0.9864044189 NSEED= 64645
RANDF(0)= 0.5138092041 NSEED= 33673

average= 0.4990450740 of 10000 randoms

NSTEP= 0 MYPRES= 1 : *** PINIT, c1s13.f: MDPART OK ***
IMAXPT= 21504 .LE. MDPART= 520000
NSTEP= 0 MYPRES= 1 : *** PINIT, c1s13.f: MDCA OK ***
INDLPA 73 .LE. MDCA= 82000

LOAD BALANCE ANALYSIS

2 Processes each have 4 blocks

TOTALS

Processes 2 should be 2
Blocks 8 should be 8

Timer Overhead, T0 = 0.151040000000435271E-04

```
=====
===          LPM3 BENCHMARK          ===
=====
===
=== UKAEA-Culham/USAF E/M-PIC Benchmark ===
=== 3D periodic cubical electron Plasma ===
=== ----- ===
===
=== Program : LPM3: Local Partical Mesh ===
=== Version : PVM + Fortran 77          ===
=== Author  : Roger W. Hockney          ===
=== Code Opt: Roger W. Hockney          ===
=== Update  : Oct, 1994; Release: 1.0    ===
===                                     ===
=====
```

Run on 400 node IBM SP2 at USAF/DOD
Maui High Performance Computer Center

Using thin nodes with 64MByte memory

Benchmark was Roger Hockney

95/09/12
14:10:28

README-LPM3

3

The calculation proceeds ...Please, wait.

```
Benchmark run started, step = 0
Beg Step = 1 mypres = 1 parts = 0 mesh = 0 nseed = 15953 itot= 0 ileave= 0
Beg Step = 2 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 55
Beg Step = 3 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 71
Beg Step = 4 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 50
Beg Step = 5 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 53
Beg Step = 6 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 58
Beg Step = 7 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 61
Beg Step = 8 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 55
Beg Step = 9 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 74
Beg Step = 10 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 52
Beg Step = 20 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 53
Beg Step = 30 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 55
Beg Step = 40 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 62
Beg Step = 50 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 46
Beg Step = 60 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 59
Beg Step = 70 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 58
Beg Step = 80 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 512 ileave= 53
Beg Step = 90 mypres = 1 parts = 2047 mesh = 960 nseed = 15953 itot= 512 ileave= 54
Beg Step = 100 mypres = 1 parts = 2048 mesh = 960 nseed = 15953 itot= 511 ileave= 64
End Step = 100 mypres = 1 parts = 2051 mesh = 960 nseed = 15953 itot= 512 ileave= 53
Benchmark run ended, step = 100
```

```
*****
LPM3 - VALIDATION CHECK
*****
```

Total number particles all processes:

this run = 4096

Reference number is = 4096

percent difference = 0.0000000000E+00 %

Total number mesh-points all processes:

this run = 1920

Reference number is = 1920

```
*****
DIFFERENCE < 10%, VALIDATION OK
*****
```

```
*****
LPM3 - RESULT SUMMARY
*****
```

Number processes, NPRES = 2

Number blocks, NBLOCK = 8

Number elements, INELEM = 512

Number patches, NPATCH = 72

Input file, CHREFN =neut50

Output files =o_neut50p2:<n> for proc <n>

MXPASW=2: per-processor code

Communications analysis:

Elapsed (XPATCH) wall time = 7.802058697 s
(17.04426765 %)

Total number CALLs to XPATCH = 800

Total number exchanges(CSEND+CRECV) = 800

Total byte CSEND = 725200 Byte

Total byte CRECV = 725308 Byte

Average CSEND length = 906.5000000 Byte

Average CRECV length = 906.6350098 Byte

Time per exchange = 9752.573242 us

CSEND bandwidth = 0.9294982255E-01 MByte/s

Number of Timesteps this run = 100

Elapsed Benchmark wall time = 45.77526474 s

Temporal Performance = 2.184586048 tstep/s

95/09/12
14:10:28

README-LPM3

4

LOAD BALANCE ANALYSIS

2 Processes each have 4 blocks

TOTALS

Processes 2 should be 2
Blocks 8 should be 8

Elapsed time this run = 46.907288 sec
or = .781788 min
or = .013030 hour